UNITED STATES DISTRICT COURT
FOR THE EASTERN DISTRICT OF TEXAS
PARIS DIVISION

RECEIVED-CLERK U.S. DISTRICT COURT
03 MAY -9 PM 5:56
EASTERN DISTRICT OF TEXAS

FILED
U. S. DISTRICT COURT
EASTERN DISTRICT OF TEXAS
MAY 9 2003
DAVID MALAND, CLERK
By
Deputy MM

TEXAS INSTRUMENTS INCORPORATED

Plaintiff,

v.

INTERGRAPH CORPORATION, and Z/I IMAGING CORPORATION

Defendants,

CIVIL ACTION NO. 3:03CV21 LD

JURY TRIAL DEMANDED

COMPLAINT AND JURY DEMAND

Plaintiff Texas Instruments Incorporated, by and through the undersigned attorneys, hereby files this Complaint against Defendants Intergraph Corporation and Z/I Imaging Corporation requesting damages and injunctive relief upon personal knowledge as to its own acts and circumstances and upon information and belief as to the acts and circumstances of others as follows:

THE PARTIES

1. Plaintiff Texas Instruments Incorporated ("TI") is a corporation incorporated under the laws of the State of Delaware and has its principal place of business at 12500 TI Boulevard, Dallas, Texas 75243-4136.

2. Defendant Intergraph Corporation is a corporation incorporated under the laws of the State of Delaware and has its principal place of business in Huntsville, Alabama.

3. Defendant Z/I Imaging Corporation is a corporation incorporated under the laws of the State of Delaware and has its principal place of business in Huntsville, Alabama. Z/I Imaging Corporation is a wholly owned subsidiary of Intergraph Corporation.

JURISDICTION AND VENUE

4. This action arises under the Patent Laws of the United States, 35 U.S.C. § 101 *et seq.* and is being brought to redress the infringement of United States Patent Nos. 5,862,394 ("the '394 patent"), 6,173,409 B1 ("the '409 patent"), and 6,397,340 B2 ("the '340 patent"), all of which are owned by TI. Copies of the '394, '409, and '340 patents are attached hereto as Exhibits 1, 2, and 3, respectively. Accordingly, subject matter jurisdiction over this Complaint is conferred upon this Court pursuant to 28 U.S.C. §§ 1331 and 1338.

5. Defendants Intergraph Corporation and Z/I Imaging Corporation (collectively referred to herein as "Intergraph") regularly conduct business in this district and have committed acts of infringement of the '394, '409, and '340 patents within this judicial district. Accordingly, this Court has personal jurisdiction over Intergraph, and venue is proper under 28 U.S.C. §§ 1391(b), (c) and 1400(b).

COUNT ONE

INFRINGEMENT OF U.S. PATENT NO. 5,862,394

6. TI incorporates and realleges all of the foregoing paragraphs as if fully set forth herein.

7. On March 21, 1996, LaVaughn F. Watts and William F. Jergens filed an application for a United States patent directed to an "Electronic Apparatus Having a Software Controlled Power Switch." On January 19, 1999, the United States Patent and Trademark Office duly and legally issued the Watts *et al.* application as the '394 patent.

8. TI is the owner of all right, title, and interest in and to the invention of the '394 patent by assignment.

9. Intergraph is directly infringing, contributing to the infringement of, or inducing others to infringe the '394 patent, by making, using, offering to sell, or selling, within this judicial district and elsewhere, products including the Intergraph Video Analyst System, Z/I ImageStation, LSTD-R Command and Display Console, TD-R 2030N Series NEMA Workstation, and TD-R 2030R Series Rack Mountable Workstation.

10. Intergraph has profited through infringement of the '394 patent. As a result of Intergraph's unlawful infringement of the '394 patent, TI has suffered, and will continue to suffer, grievous damage.

11. Intergraph's acts of infringement are made with full knowledge of TI's rights in the '394 patent. Such acts constitute willful infringement and make this case exceptional pursuant to 35 U.S.C. §§ 284 and 285 and entitle TI to enhanced damages and reasonable attorneys' fees.

12. Intergraph will continue its unlawful infringing activity unless enjoined by this Court.

COUNT TWO

INFRINGEMENT OF U.S. PATENT NO. 6,173,409 B1

13. TI incorporates and realleges all of the foregoing paragraphs as if fully set forth herein.

14. On September 8, 1999, LaVaughn F. Watts, Jr. and Steven J. Wallace filed a continuation application for a United States patent directed to a "Real-Time Power Conservation

for Electronic Device Having a Processor." On January 9, 2001, the United States Patent and Trademark Office duly and legally issued the Watts, Jr. *et al.* application as the '409 patent.

15. TI is the owner of all right, title, and interest in and to the invention of the '409 patent by assignment.

16. Intergraph is directly infringing, contributing to the infringement of, or inducing others to infringe the '409 patent, by making, using, offering to sell, or selling, within this judicial district and elsewhere, products including the Intergraph Video Analyst System, Z/I ImageStation, LSTD-R Command and Display Console, TD-R 2030N Series NEMA Workstation, and TD-R 2030R Series Rack Mountable Workstation.

17. Intergraph has profited through infringement of the claims of the '409 patent. As a result of Intergraph's unlawful infringement of the '409 patent, TI has suffered, and will continue to suffer, grievous damage.

18. Intergraph's acts of infringement are made with full knowledge of TI's rights in the '409 patent. Such acts constitute willful infringement and make this case exceptional pursuant to 35 U.S.C. §§ 284 and 285 and entitle TI to enhanced damages and reasonable attorneys' fees.

19. Intergraph will continue its unlawful infringing activity unless enjoined by this Court.

COUNT THREE

INFRINGEMENT OF U.S. PATENT NO. 6,397,340 B2

20. TI incorporates and realleges all of the foregoing paragraphs as if fully set forth herein.

21. On January 9, 2001, LaVaughn F. Watts, Jr. and Steven J. Wallace filed a continuation application for a United States patent directed to a "Real-Time Power Conservation for Electronic Device Having a Processor." On May 28, 2002, the United States Patent and Trademark Office duly and legally issued the Watts, Jr. *et al.* application as the '340 patent.

22. TI is the owner of all right, title, and interest in and to the invention of the '340 patent by assignment.

23. Intergraph is directly infringing, contributing to the infringement of, or inducing others to infringe the '340 patent, by making, using, offering to sell, and selling, within this judicial district and elsewhere, products including the Intergraph Video Analyst System, Z/I ImageStation, LSTD-R Command and Display Console, TD-R 2030N Series NEMA Workstation, and TD-R 2030R Series Rack Mountable Workstation.

24. Intergraph has profited through infringement of the claims of the '340 patent. As a result of Intergraph's unlawful infringement of the '340 patent, TI has suffered, and will continue to suffer, grievous damage.

25. Intergraph's acts of infringement are made with full knowledge of TI's rights in the '340 patent. Such acts constitute willful infringement and make this case exceptional pursuant to 35 U.S.C. §§ 284 and 285 and entitle TI to enhanced damages and reasonable attorneys' fees.

26. Intergraph will continue its unlawful infringing activity unless enjoined by this Court.

JURY DEMAND

27. TI requests a trial by jury of all claims so triable.

PRAYER FOR RELIEF

WHEREFORE, TI prays that this Court enter judgment:

1. That Defendants have infringed U.S. Patent Nos. 5,862,394, 6,173,409 B1, and 6,397,340 B2;
2. That Defendants' infringement is willful;
3. That this is an exceptional case;
4. Permanently enjoining and restraining Defendants and their respective agents, servants, employees, affiliates, divisions, and subsidiaries, and those in association with them, from directly or indirectly infringing U.S. Patent Nos. 5,862,394, 6,173,409 B1, and 6,397,340 B2;
5. Awarding TI damages for Defendants' infringement including costs and pre- and post-judgment interest as allowed by law;
6. Awarding TI treble the amount of damages because of the willful nature of Defendants' conduct;
7. Awarding TI all costs and reasonable attorneys' fees, including interest; and
8. Granting TI such other and further relief as the Court may deem just and equitable.

Dated: May 9, 2003

Respectfully submitted,

FISH & RICHARDSON P.C.

By: Carl Roth

Thomas M. Melsheimer
Texas Bar No. 13922550
Thomas B. Walsh, IV
Texas Bar No. 00785173
Neil J. McNabnay
Texas Bar No. 24002583
1717 Main Street, Suite 5000
Dallas, TX 75201
(214) 747-5070 (Telephone)
(214) 747-2091 (Telecopy)

Carl Roth
Texas Bar No. 17312000
The Roth Law Firm
115 N. Wellington, suite 200
Marshall, TX 75670
(903) 935-1665 (Telephone)
(903) 935-1797 (Telecopy)

Glenn Perry
Texas Bar No. 15801500
Perry & Womack
P.O. Box 3266
Longview, TX 75606
(903) 757-9191 (Telephone)
(903) 758-3239 (Telecopy)

Of Counsel:
Linda Kordziel
FISH & RICHARDSON P.C.
1425 K Street, N.W., 11th Floor
Washington, DC 20005
(202) 783-5070 (Telephone)
(202) 783-2331 (Telecopy)

Counsel for Plaintiff
TEXAS INSTRUMENTS INCORPORATED

EXHIBIT 1

US005862394A

United States Patent [19]

Watts et al.

[11] **Patent Number: 5,862,394**

[45] **Date of Patent: Jan. 19, 1999**

[54] **ELECTRONIC APPARATUS HAVING A SOFTWARE CONTROLLED POWER SWITCH**

[75] Inventors: **LaVaughn F. Watts**, Temple; **William F. Jergens**, Belton, both of Tex.

[73] Assignee: **Texas Instruments Incorporated**, Dallas, Tex.

[21] Appl. No.: **621,741**

[22] Filed: **Mar. 21, 1996**

[51] **Int. Cl.**[6] **G06F 1/24**; G06F 1/26

[52] **U.S. Cl.** **395/750.07**; 395/750.01

[58] **Field of Search** 395/750, 750.07, 395/750.01; 364/707; 365/226; 323/271

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,365,290	12/1982	Nelms et al.	364/707
4,598,383	7/1986	Leach	395/750
4,847,616	7/1989	Gotou et al.	340/825
5,167,024	11/1992	Smith et al.	395/375
5,287,525	2/1994	Lum et al.	395/750
5,450,003	9/1995	Cheon	323/272
5,504,910	4/1996	Wisor et al.	395/750
5,542,035	7/1996	Kikimis et al.	395/750
5,598,567	1/1997	Ninomiya	395/750

FOREIGN PATENT DOCUMENTS

0730217A1	9/1996	European Pat. Off.	G06F 1/32
2295040	2/1995	United Kingdom	G06F 1/30

OTHER PUBLICATIONS

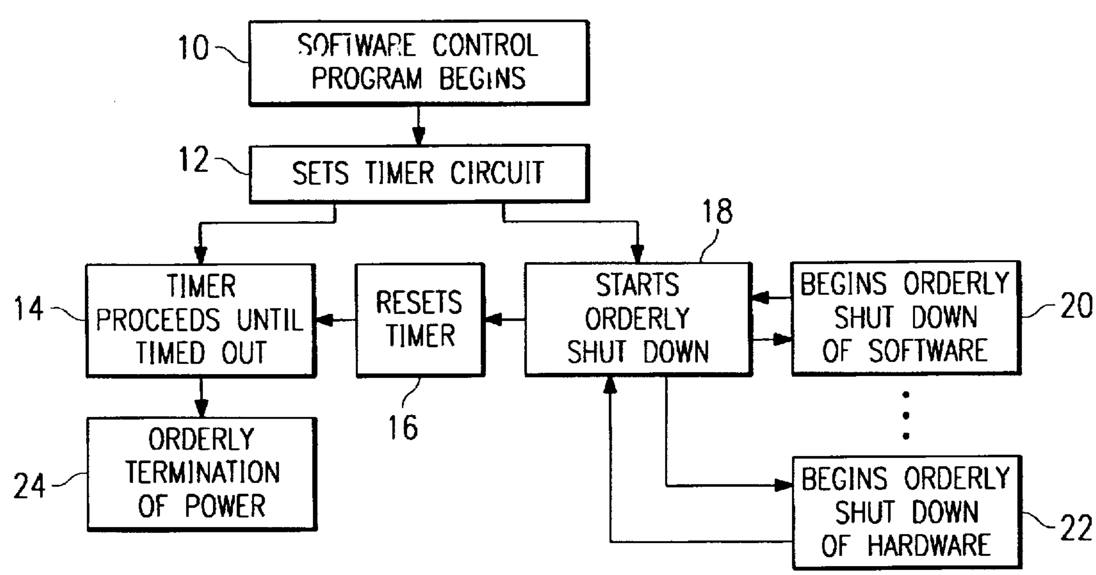
"Method for Determining a Low Battery Condition," IBM Technical Bulletin, vol. 38, No. 6, Jun. 1, 1995 pp. 295–296.

Primary Examiner—Gopal C. Ray
Attorney, Agent, or Firm—Ronald O. Neerings; James C. Kesterson; Richard L. Donaldson

[57] **ABSTRACT**

This is a system and method of intelligently terminating power to a computing device. The system may comprise: a processing device; a power source connected to the processing device; a switch connected to the power source; and a control system run by the processing device and connected to the power source and the switch. In addition, the system may include a deadman timer which provides a fail-safe operation. Further, the system may include a method and apparatus for executing an orderly shut down procedure for software and hardware. Moreover, the system could be tied to a thermal and/or power management system. Additionally, the system could initiate an orderly shut down of peripheral devices connected to the system serially or by parallel connections. Other devices, systems and methods are also described.

59 Claims, 119 Drawing Sheets



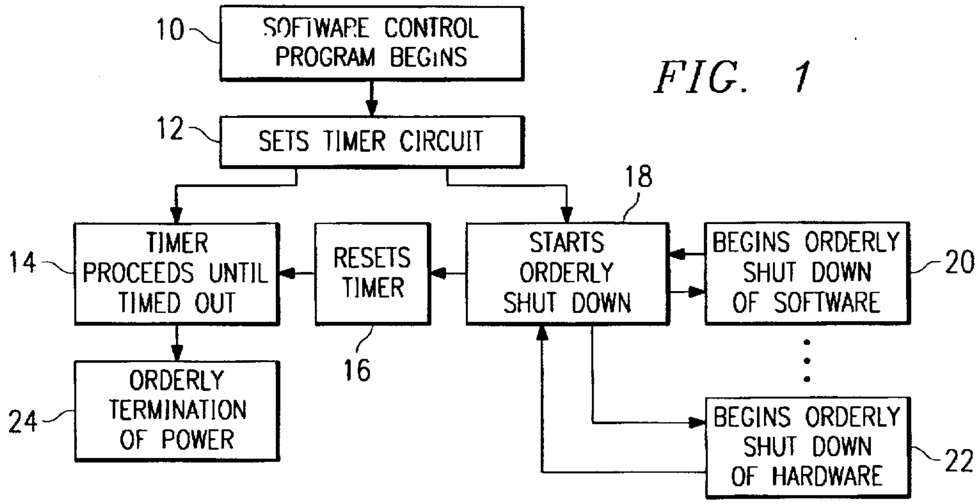


FIG. 1

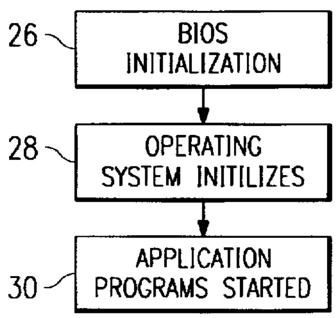


FIG. 2

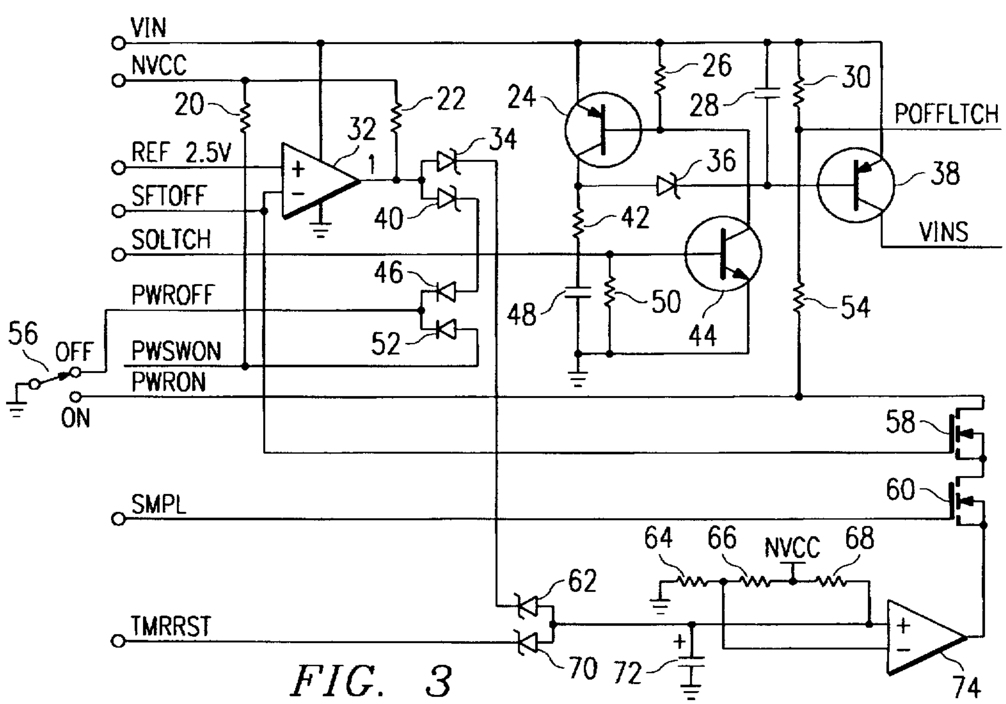
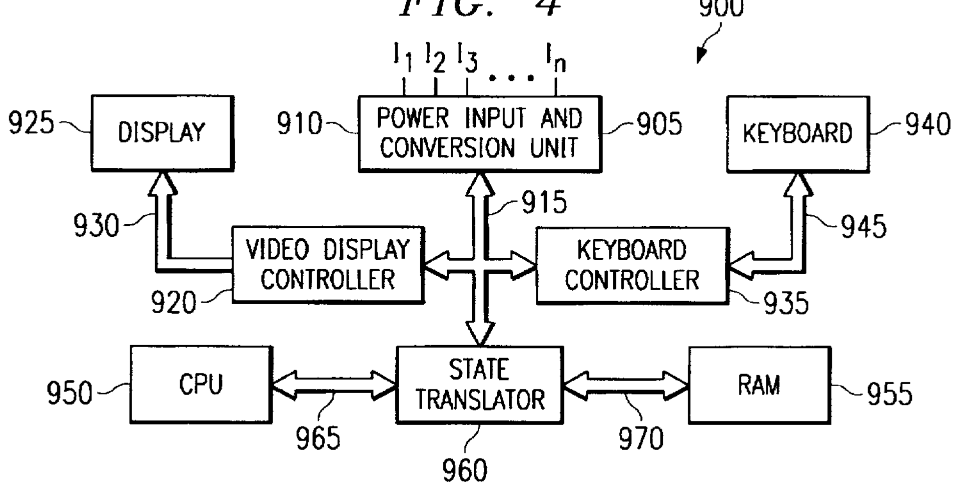


FIG. 3

FIG. 4


900
I_1 I_2 I_3 . . . I_n
925
DISPLAY
910
POWER INPUT AND CONVERSION UNIT
905
KEYBOARD
940
930
915
945
VIDEO DISPLAY CONTROLLER
KEYBOARD CONTROLLER
920
935
950
CPU
STATE TRANSLATOR
RAM
955
965
970
960

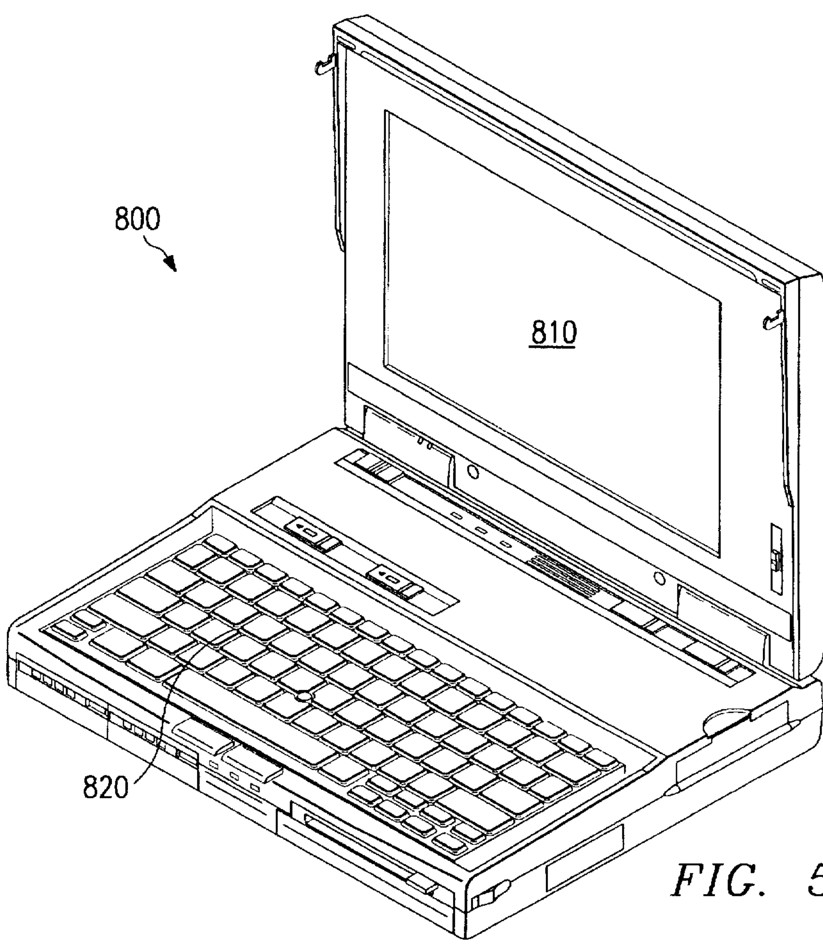

800
810
820

FIG. 5

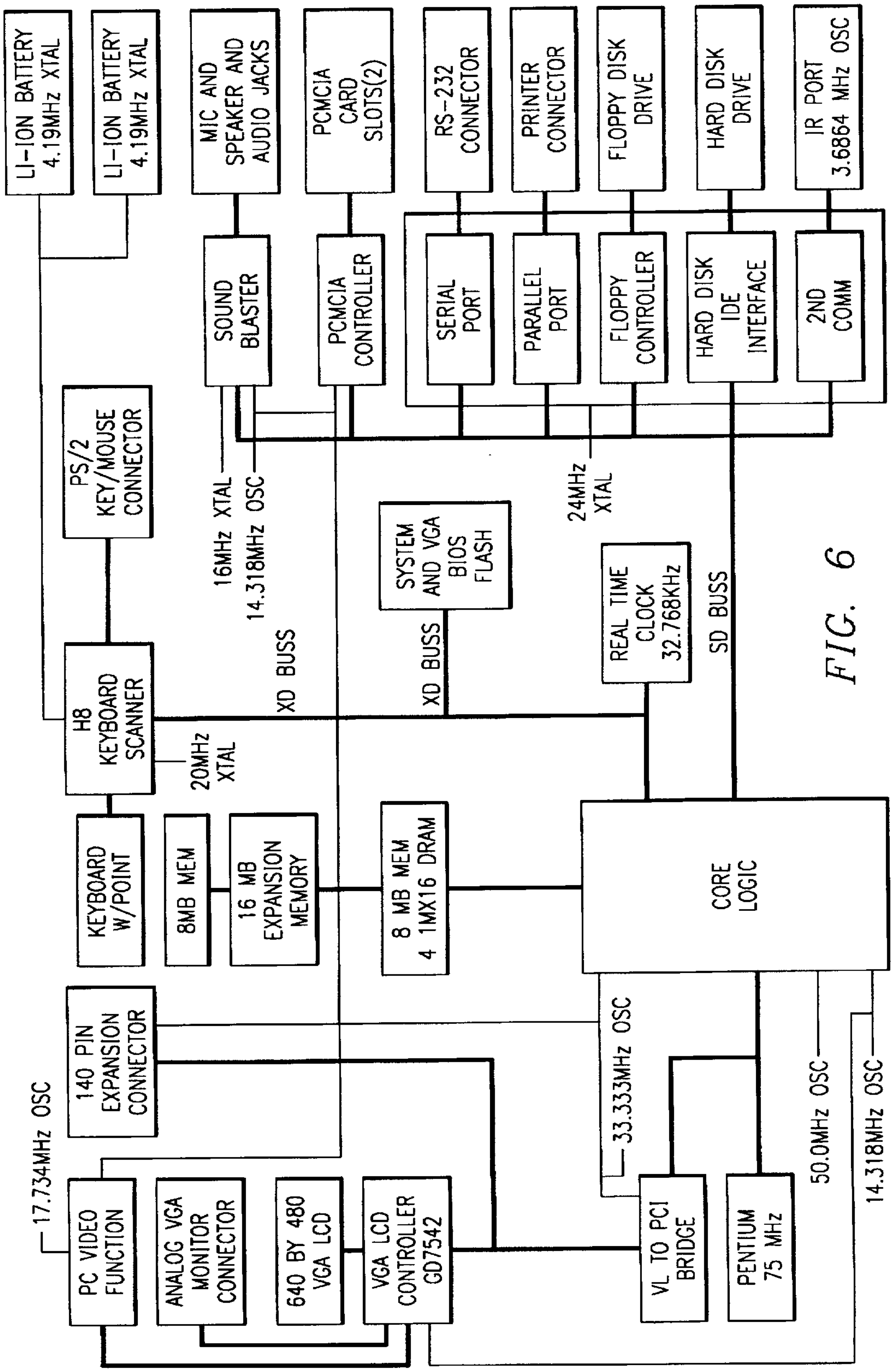

LI-ION BATTERY 4.19MHz XTAL
LI-ION BATTERY 4.19MHz XTAL
MIC AND SPEAKER AND AUDIO JACKS
PCMCIA CARD SLOTS(2)
RS-232 CONNECTOR
PRINTER CONNECTOR
FLOPPY DISK DRIVE
HARD DISK DRIVE
IR PORT 3.6864 MHz OSC
SOUND BLASTER
PCMCIA CONTROLLER
SERIAL PORT
PARALLEL PORT
FLOPPY CONTROLLER
HARD DISK IDE INTERFACE
2ND COMM
PS/2 KEY/MOUSE CONNECTOR
16MHz XTAL
14.318MHz OSC
24MHz XTAL
SYSTEM AND VGA BIOS FLASH
REAL TIME CLOCK 32.768KHz
SD BUSS
XD BUSS
XD BUSS
H8 KEYBOARD SCANNER
20MHz XTAL
KEYBOARD W/POINT
8MB MEM
16 MB EXPANSION MEMORY
8 MB MEM 4 1MX16 DRAM
CORE LOGIC
140 PIN EXPANSION CONNECTOR
17.734MHz OSC
33.333MHz OSC
50.0MHz OSC
14.318MHz OSC
PC VIDEO FUNCTION
ANALOG VGA MONITOR CONNECTOR
640 BY 480 VGA LCD
VGA LCD CONTROLLER GD7542
VL TO PCI BRIDGE
PENTIUM 75 MHz

FIG. 6

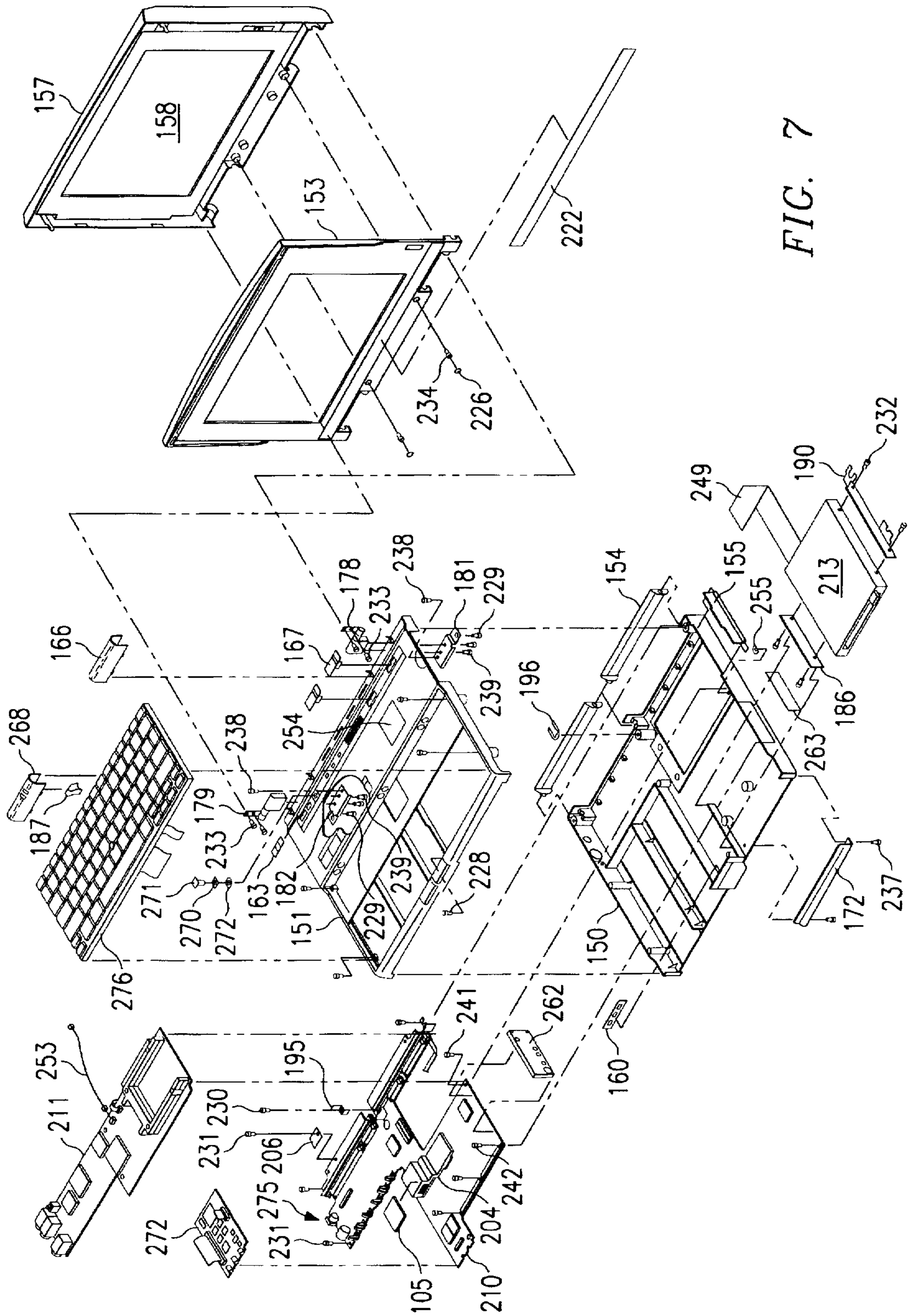

157
158
153
222
234
226
232
190
249
213
255
155
154
186
263
229
181
238
233
178
167
166
239
196
254
238
179
268
187
233
271
270
272
163
182
239
229
228
151
150
172
237
276
241
262
160
253
211
195
230
231
206
275
231
272
242
204
105
210

FIG. 7

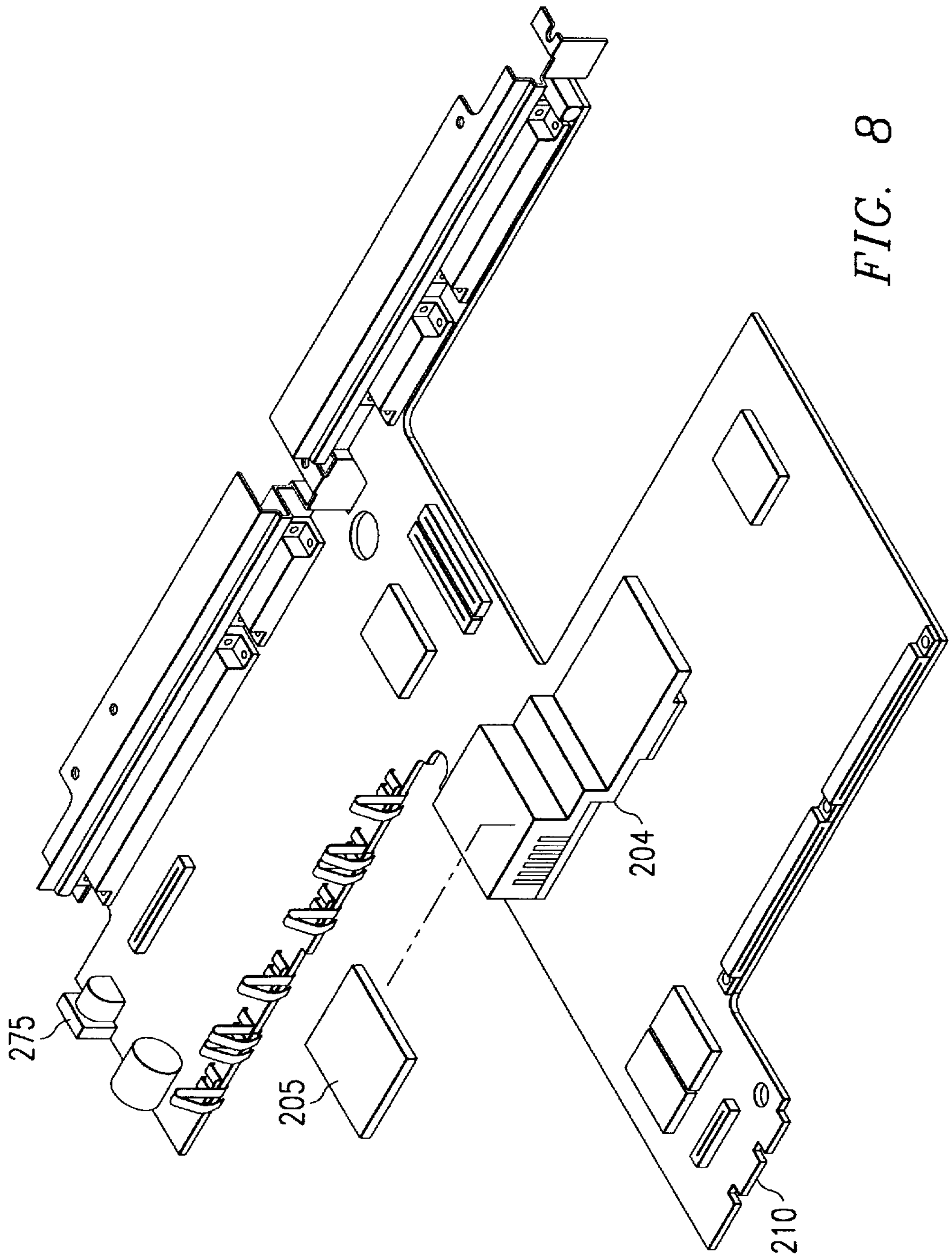


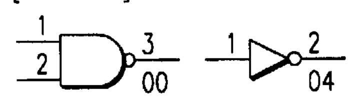
FIG. 8

NOTES : UNLESS OTHERWISE SPECIFIED :

1. ALL IC DEVICE TYPES ARE PREFIXED WITH SN74.
2. THE FOLLOWING PREFIX'S ARE ALWAY'S USED:
   T IS EQUAL TO "LS"
   AT IS EQUAL TO "ALS"
3. THE FOLLOWING PREFIX'S ARE USED ONLY WHEN INSUFFICIENT CHARACTERS ARE AVAILABLE:
   A IS EQUAL TO "ACT"
   B IS EQUAL TO "BCT"
   V IS EQUAL TO "AS"
   W IS EQUAL TO "AT" OR "ALS"
4. IC PACKAGE TYPE IS INDICATED BY THE FOLLOWING SUFFIX'S:

DUAL-IN-LINE, PLASTIC	= "N" OR BLANK
DUAL-IN-LINE, PLASTIC [WIDE]	= NW
DUAL-IN-LINE, CERAMIC	= J
DUAL-IN-LINE, CERAMIC [WIDE]	= JD
CHIP CARRIER, PLASTIC	= F
CHIP CARRIER IN A S.M. SCKT	= FF
CHIP CARRIER IN A PGA SCKT	= FX
CHIP CARRIER, CERAMIC [RECT]	= FE
CHIP CARRIER, CERAMIC [SQUARE]	= FH
FLAT PACKAGE, CERAMIC	= U
FLAT PACKAGE, CERAMIC [WIDE]	= W
GRID ARRAY, PLASTIC	= X
GRID ARRAY, PLASTIC [LIF SCKT]	= XL
GRID ARRAY, PLASTIC [ZIF SCKT]	= XZ
GRID ARRAY, CERAMIC	= Y
GRID ARRAY, CERAMIC [LIF SCKT]	= YL
GRID ARRAY, CERAMIC [ZIF SCKT]	= YZ
SINGLE-IN-LINE	= E,L,M,G
"SOIC", PLASTIC	= D
"SOIC", PLASTIC [WIDE]	= DW
"SOJ", PLASTIC, J LEADS	= R

5. VCC IS APPLIED TO PIN 8 OF ALL 8-PIN IC's, PIN 14 OF ALL 14-PIN IC's, PIN 16 OF ALL 16-PIN IC's, PIN 20 OF ALL 20-PIN IC's, ETC.
6. GROUND IS APPLIED TO PIN 4 OF ALL 8-PIN IC's, PIN 7 OF ALL 14-PIN IC's, PIN 8 OF ALL 16-PIN IC's, PIN 10 OF ALL 20-PIN IC's, ETC.
7. DEVICE TYPE, PIN NUMBERS, AND REFERENCE DESIGNATOR [LOCATION] OF GATES ARE SHOWN AS FOLLOWS:



   00 AND 04 = DEVICE TYPES
   1, 2, AND 3 = PIN NUMBERS
   U01 AND U02 = REF. DESIGNATOR [LOCATION]
8. RESISTANCE VALUES ARE IN OHMS.
9. RESISTORS ARE 1/8 WATT, 5%.
10. CAPACITANCE VALUES ARE IN MICROFARADS.
11. CAPACITORS ARE 50V, 10%.
12. THIS COUPON WILL BE USED ON ALL COMMERICAL MULTILAYER BOARDS.

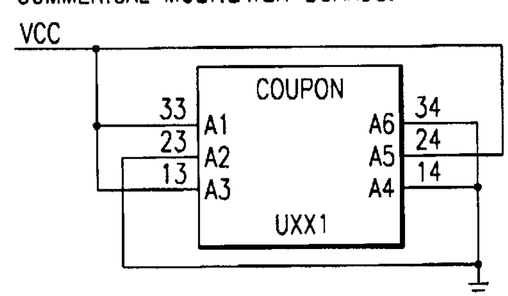


FIG. 9

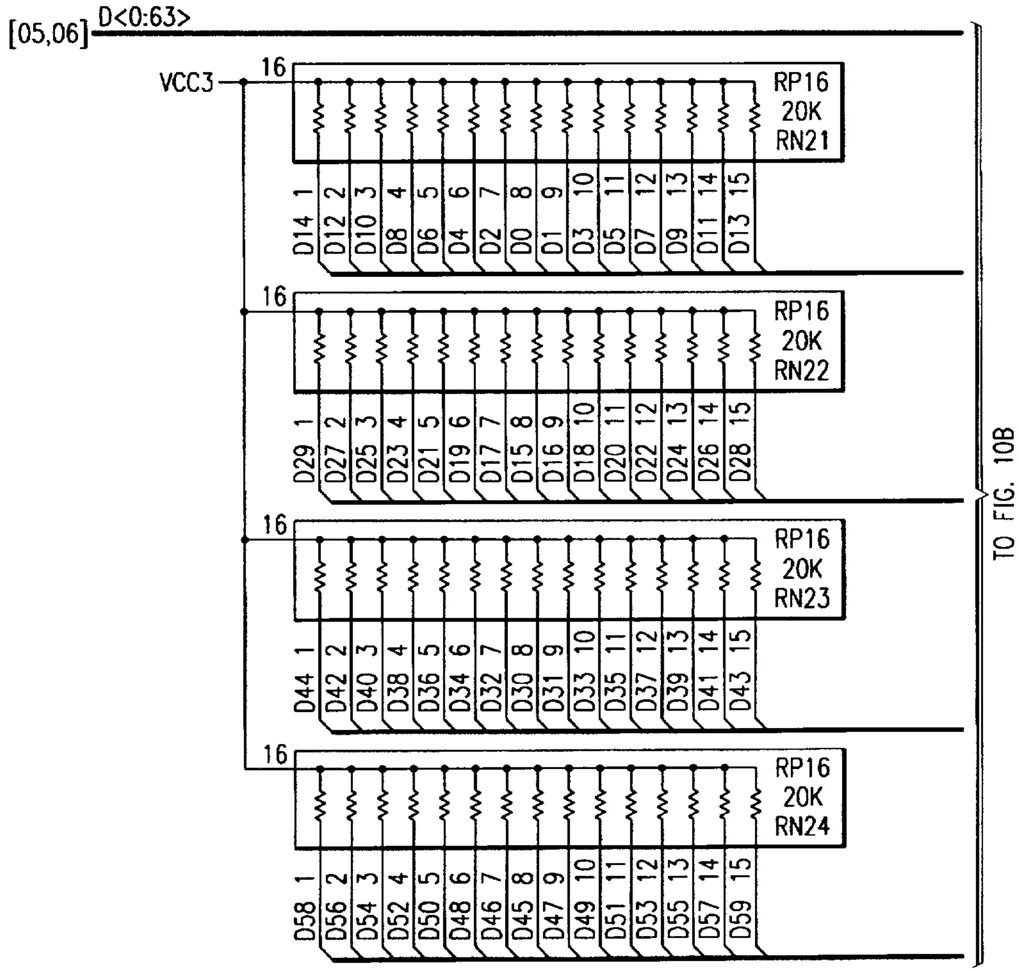
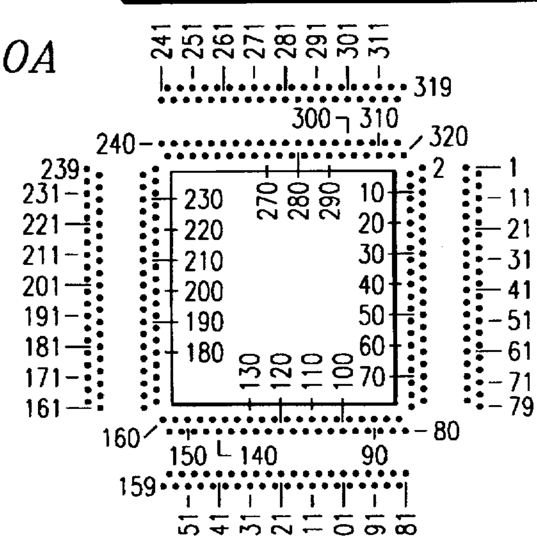


FIG. 10A



PENTIUM PINOUT TCP 320
BACKSIDE VIEW (TOP OF PWB)

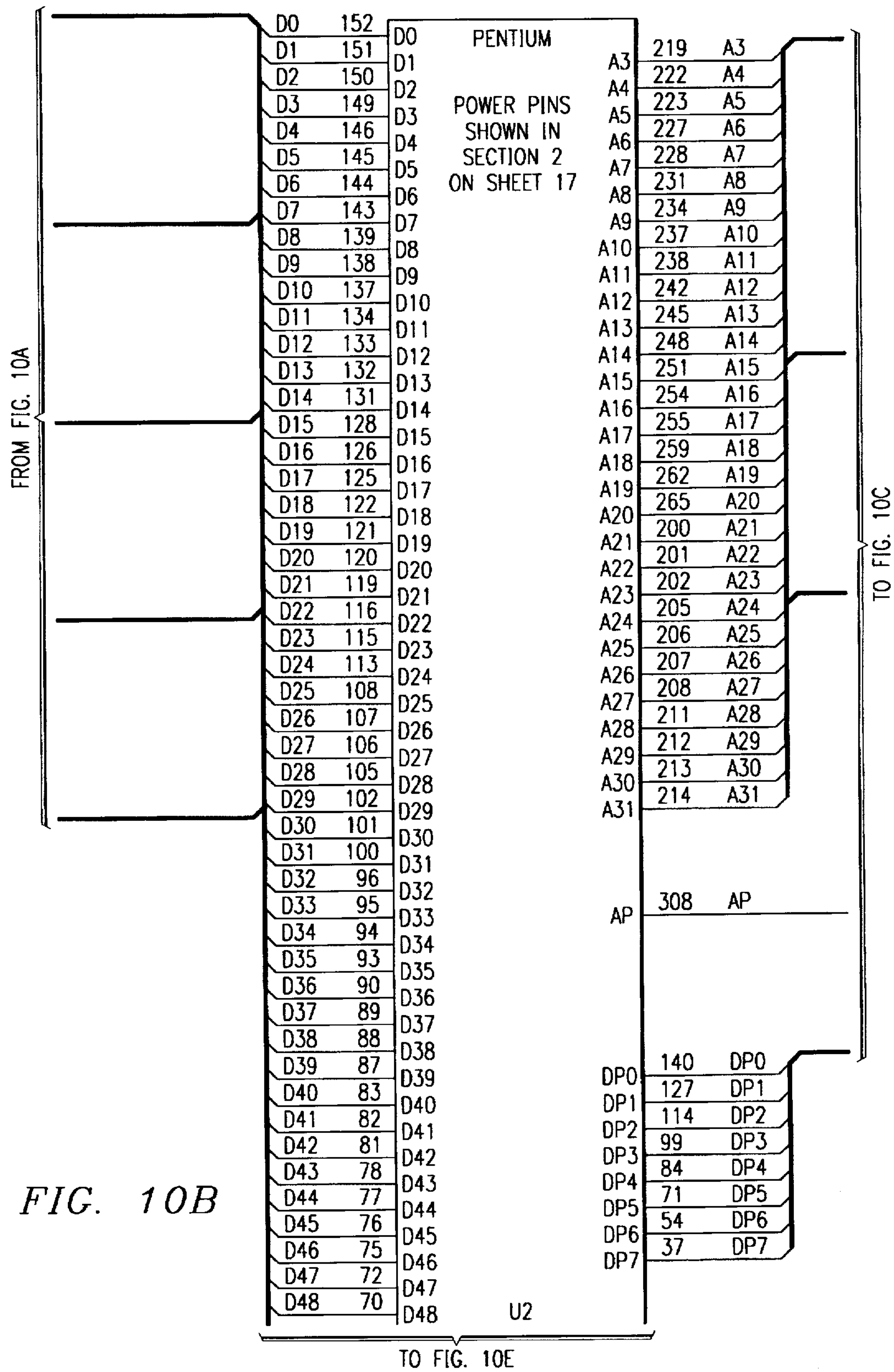

FROM FIG. 10A
PENTIUM
POWER PINS SHOWN IN SECTION 2 ON SHEET 17
U2
TO FIG. 10C
TO FIG. 10E
AP
308

FIG. 10B

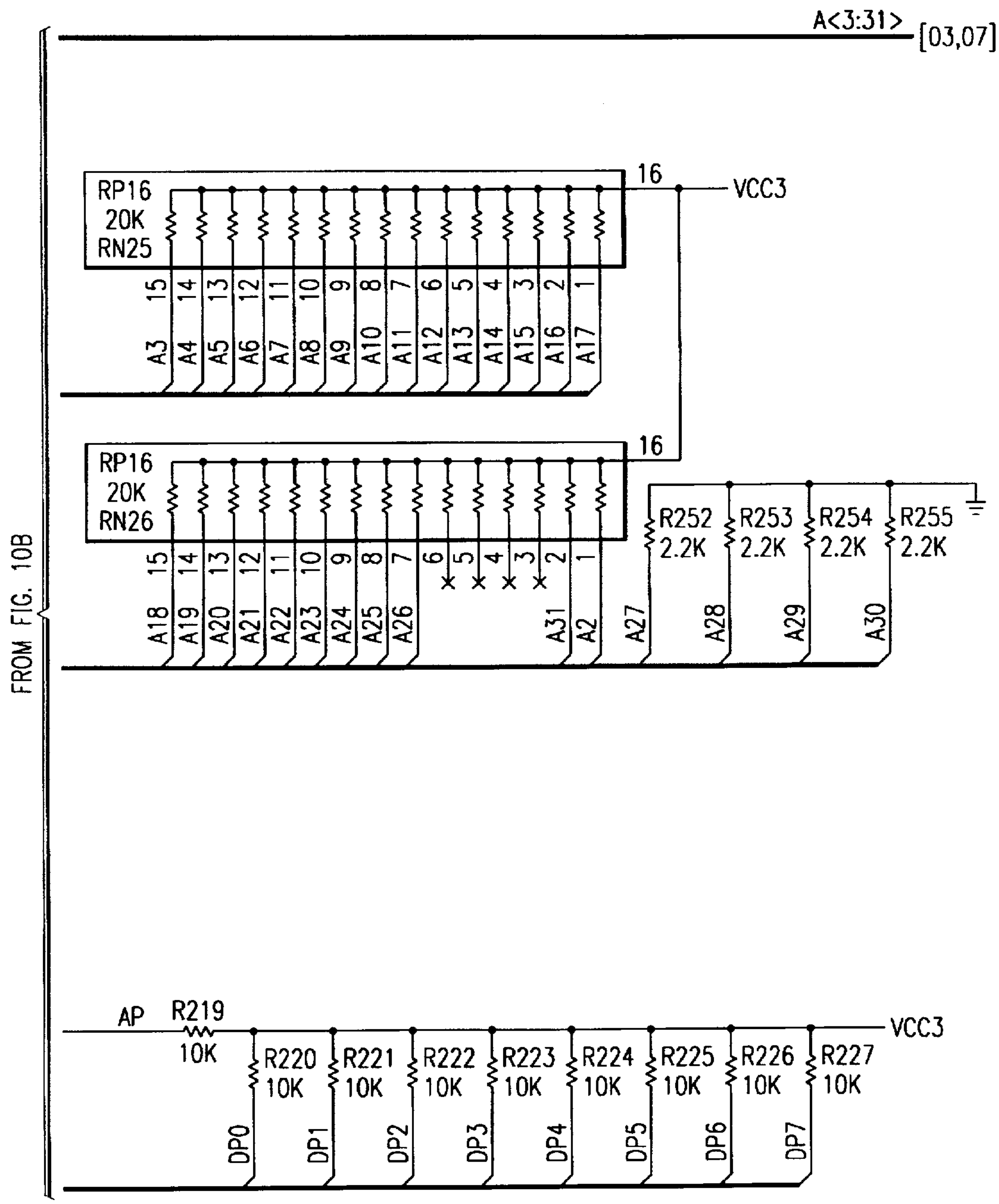

A<3:31>
[03,07]
FROM FIG. 10B
RP16
20K
RN25
16
VCC3
A3
A4
A5
A6
A7
A8
A9
A10
A11
A12
A13
A14
A15
A16
A17
RP16
20K
RN26
16
A18
A19
A20
A21
A22
A23
A24
A25
A26
A31
A2
A27
A28
A29
A30
R252
2.2K
R253
2.2K
R254
2.2K
R255
2.2K
AP
R219
10K
VCC3
R220
10K
R221
10K
R222
10K
R223
10K
R224
10K
R225
10K
R226
10K
R227
10K
DP0
DP1
DP2
DP3
DP4
DP5
DP6
DP7

FIG. 10C

FIG. 10D

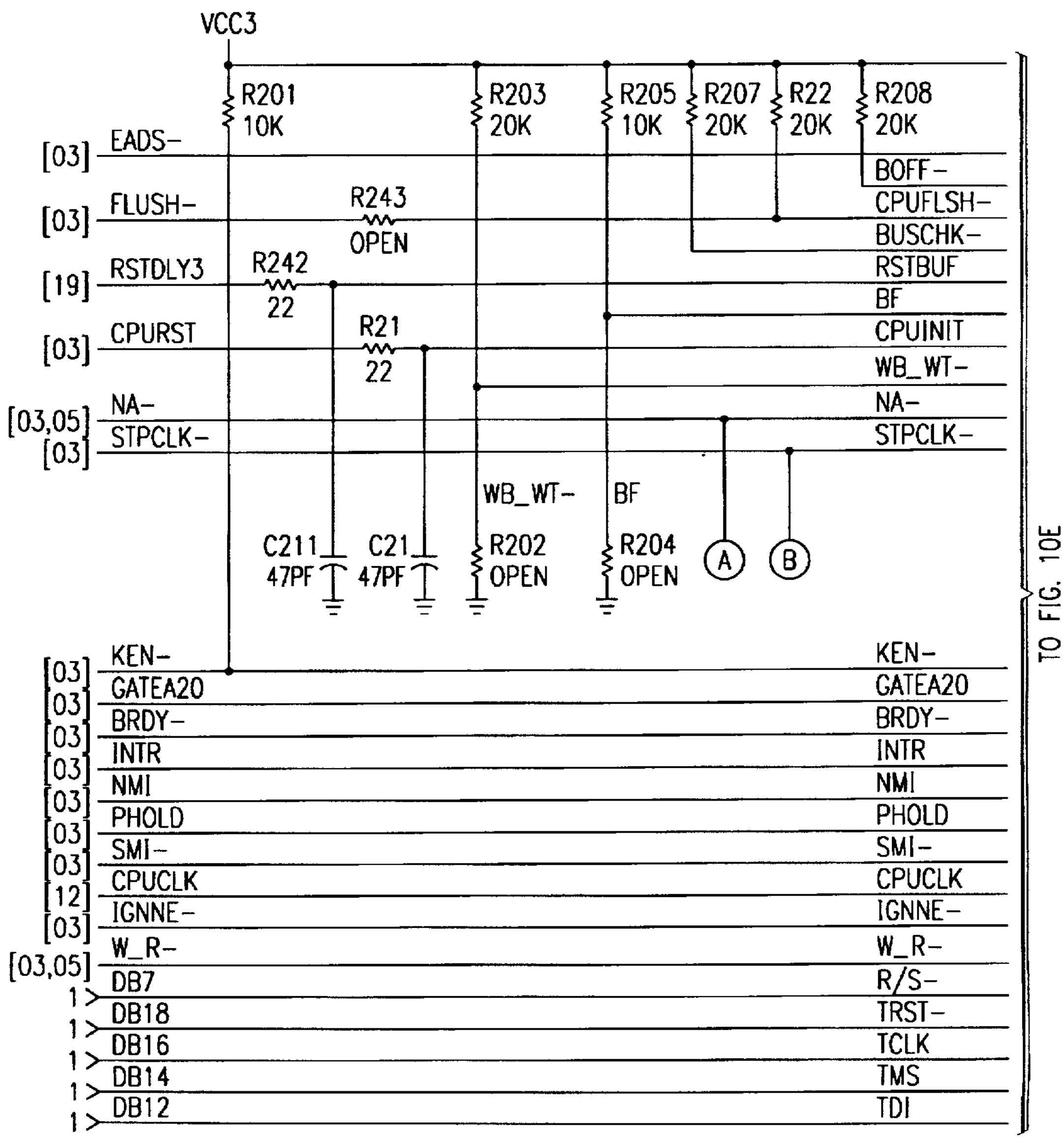
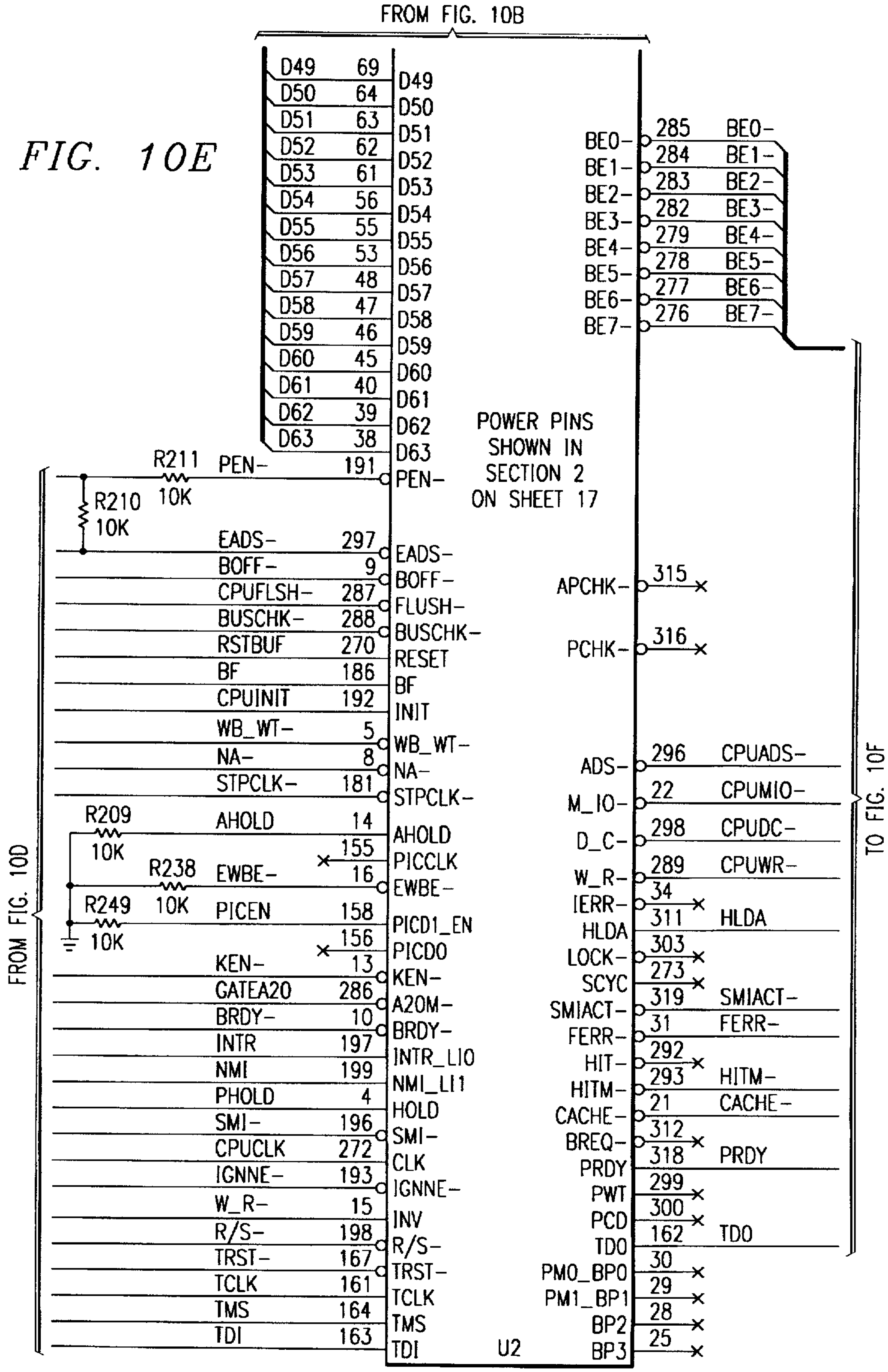
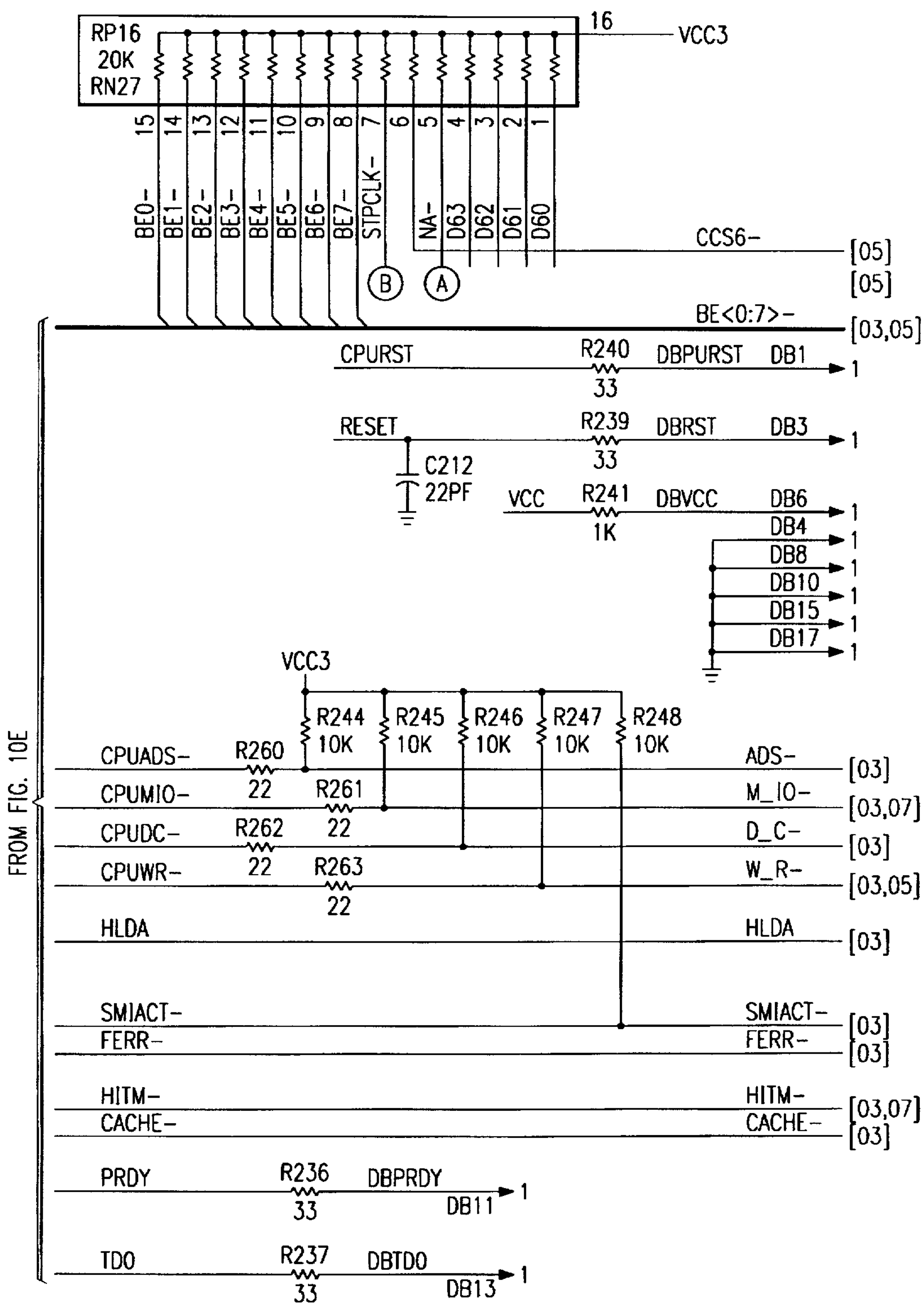

VCC3
R201 10K
R203 20K
R205 10K
R207 20K
R22 20K
R208 20K
[03] EADS-
[03] FLUSH-
R243 OPEN
[19] RSTDLY3
R242 22
[03] CPURST
R21 22
[03,05] NA-
[03] STPCLK-
BOFF-
CPUFLSH-
BUSCHK-
RSTBUF
BF
CPUINIT
WB_WT-
NA-
STPCLK-
WB_WT-
BF
C211 47PF
C21 47PF
R202 OPEN
R204 OPEN
A
B
TO FIG. 10E
[03] KEN-
[03] GATEA20
[03] BRDY-
[03] INTR
[03] NMI
[03] PHOLD
[03] SMI-
[12] CPUCLK
[03] IGNNE-
[03,05] W_R-
1 DB7
1 DB18
1 DB16
1 DB14
1 DB12
KEN-
GATEA20
BRDY-
INTR
NMI
PHOLD
SMI-
CPUCLK
IGNNE-
W_R-
R/S-
TRST-
TCLK
TMS
TDI

FIG. 10E

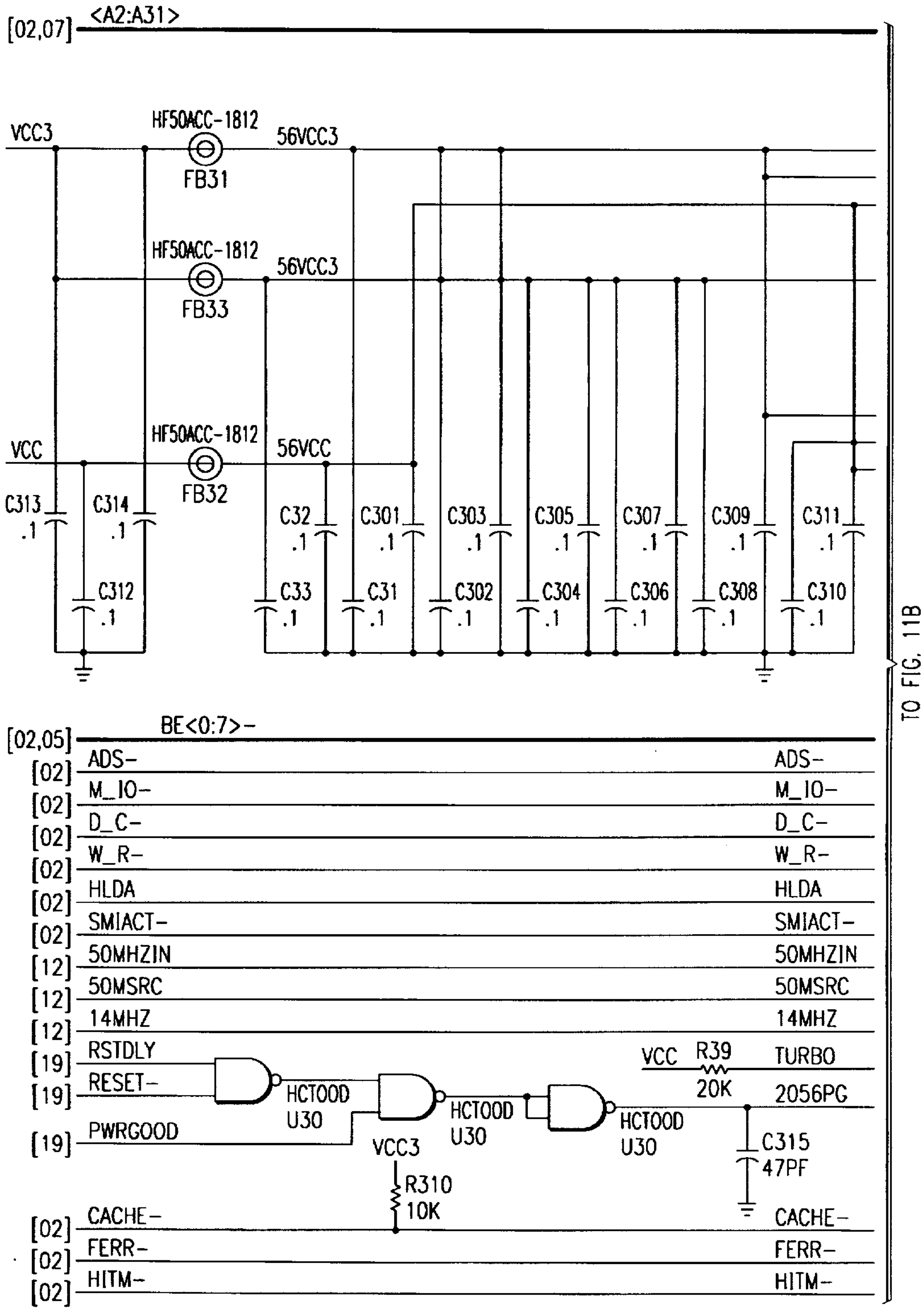


FROM FIG. 10B
D49 69 D49
D50 64 D50
D51 63 D51
D52 62 D52
D53 61 D53
D54 56 D54
D55 55 D55
D56 53 D56
D57 48 D57
D58 47 D58
D59 46 D59
D60 45 D60
D61 40 D61
D62 39 D62
D63 38 D63
POWER PINS SHOWN IN SECTION 2 ON SHEET 17
BE0- 285 BE0-
BE1- 284 BE1-
BE2- 283 BE2-
BE3- 282 BE3-
BE4- 279 BE4-
BE5- 278 BE5-
BE6- 277 BE6-
BE7- 276 BE7-
R211 10K PEN- 191 PEN-
R210 10K
EADS- 297 EADS-
BOFF- 9 BOFF-
CPUFLSH- 287 FLUSH-
BUSCHK- 288 BUSCHK-
RSTBUF 270 RESET
BF 186 BF
CPUINIT 192 INIT
WB_WT- 5 WB_WT-
NA- 8 NA-
STPCLK- 181 STPCLK-
R209 10K AHOLD 14 AHOLD
155 PICCLK
R238 10K EWBE- 16 EWBE-
R249 10K PICEN 158 PICD1_EN
156 PICD0
KEN- 13 KEN-
GATEA20 286 A20M-
BRDY- 10 BRDY-
INTR 197 INTR_LI0
NMI 199 NMI_LI1
PHOLD 4 HOLD
SMI- 196 SMI-
CPUCLK 272 CLK
IGNNE- 193 IGNNE-
W_R- 15 INV
R/S- 198 R/S-
TRST- 167 TRST-
TCLK 161 TCLK
TMS 164 TMS
TDI 163 TDI
U2
APCHK- 315
PCHK- 316
ADS- 296 CPUADS-
M_IO- 22 CPUMIO-
D_C- 298 CPUDC-
W_R- 289 CPUWR-
IERR- 34
HLDA 311 HLDA
LOCK- 303
SCYC 273
SMIACT- 319 SMIACT-
FERR- 31 FERR-
HIT- 292
HITM- 293 HITM-
CACHE- 21 CACHE-
BREQ- 312
PRDY 318 PRDY
PWT 299
PCD 300
TDO 162 TDO
PM0_BP0 30
PM1_BP1 29
BP2 28
BP3 25
FROM FIG. 10D
TO FIG. 10F



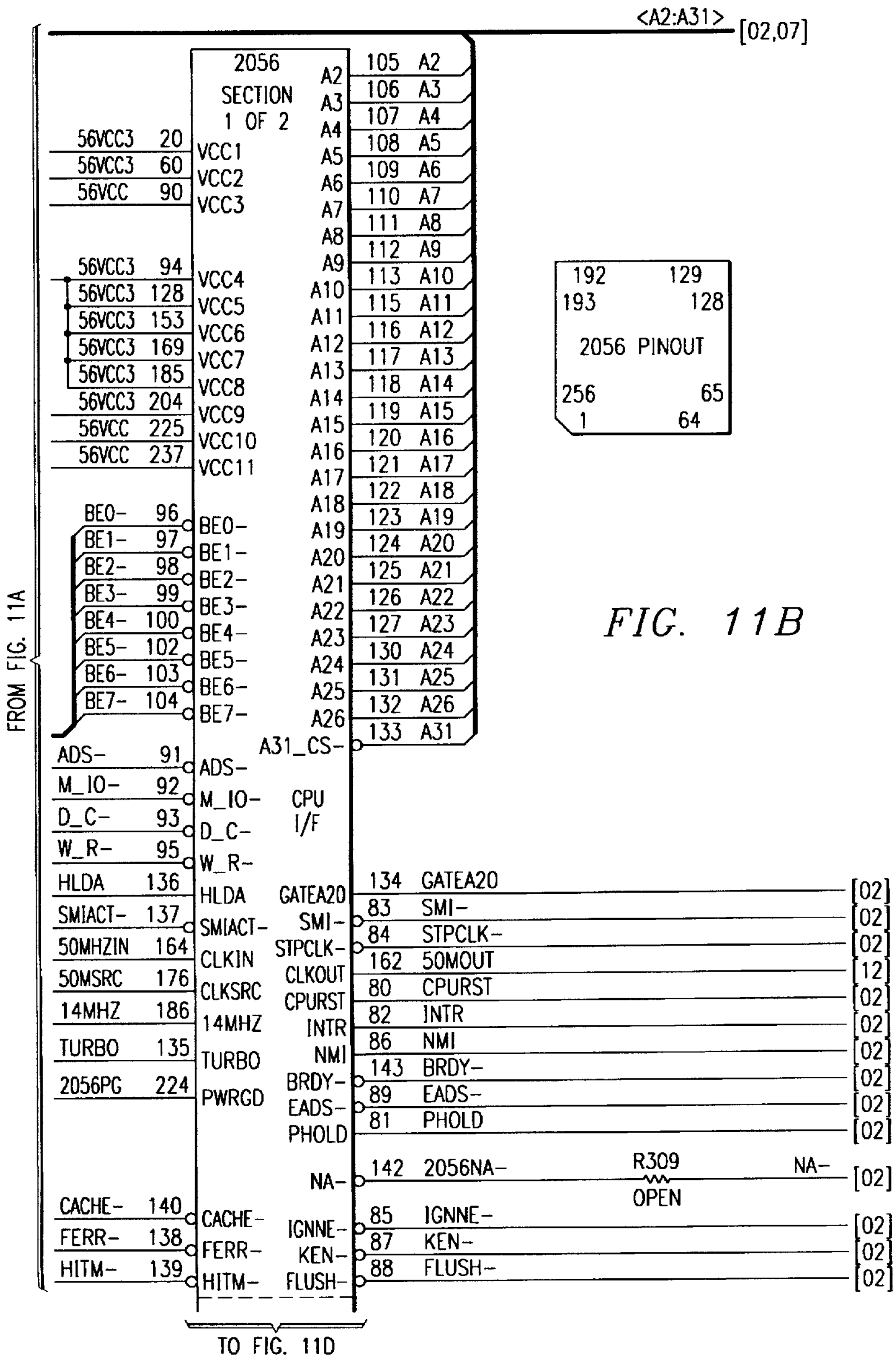
RP16
20K
RN27
16
VCC3
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
BE0- BE1- BE2- BE3- BE4- BE5- BE6- BE7- STPCLK- NA- D63 D62 D61 D60
CCS6-
[05]
[05]
B
A
BE<0:7>-
[03,05]
CPURST
R240
33
DBPURST
DB1
1
RESET
R239
33
DBRST
DB3
C212
22PF
VCC
R241
1K
DBVCC
DB6
DB4
DB8
DB10
DB15
DB17
VCC3
R244 10K
R245 10K
R246 10K
R247 10K
R248 10K
FROM FIG. 10E
CPUADS-
R260
22
ADS-
[03]
CPUMIO-
R261
22
M_IO-
[03,07]
CPUDC-
R262
22
D_C-
[03]
CPUWR-
R263
22
W_R-
[03,05]
HLDA
HLDA
[03]
SMIACT-
SMIACT-
[03]
FERR-
FERR-
[03]
HITM-
HITM-
[03,07]
CACHE-
CACHE-
[03]
PRDY
R236
33
DBPRDY
DB11
TDO
R237
33
DBTDO
DB13

FIG. 10F



[02,07]
<A2:A31>
VCC3
HF50ACC-1812
FB31
56VCC3
HF50ACC-1812
FB33
56VCC3
VCC
HF50ACC-1812
FB32
56VCC
C313 .1
C314 .1
C312 .1
C32 .1
C33 .1
C301 .1
C31 .1
C303 .1
C302 .1
C305 .1
C304 .1
C307 .1
C306 .1
C309 .1
C308 .1
C311 .1
C310 .1
TO FIG. 11B
[02,05]
BE<0:7>-
[02] ADS- ADS-
[02] M_IO- M_IO-
[02] D_C- D_C-
[02] W_R- W_R-
[02] HLDA HLDA
[02] SMIACT- SMIACT-
[12] 50MHZIN 50MHZIN
[12] 50MSRC 50MSRC
[12] 14MHZ 14MHZ
[19] RSTDLY
[19] RESET-
[19] PWRGOOD
HCT00D U30
HCT00D U30
HCT00D U30
VCC R39 20K
TURBO
2056PG
C315 47PF
VCC3
R310 10K
[02] CACHE- CACHE-
[02] FERR- FERR-
[02] HITM- HITM-

FIG. 11A



<A2:A31> [02,07]
2056
SECTION
1 OF 2
56VCC3 20 VCC1
56VCC3 60 VCC2
56VCC 90 VCC3
56VCC3 94 VCC4
56VCC3 128 VCC5
56VCC3 153 VCC6
56VCC3 169 VCC7
56VCC3 185 VCC8
56VCC3 204 VCC9
56VCC 225 VCC10
56VCC 237 VCC11
BE0- 96 BE0-
BE1- 97 BE1-
BE2- 98 BE2-
BE3- 99 BE3-
BE4- 100 BE4-
BE5- 102 BE5-
BE6- 103 BE6-
BE7- 104 BE7-
ADS- 91 ADS-
M_IO- 92 M_IO-
D_C- 93 D_C-
W_R- 95 W_R-
HLDA 136 HLDA
SMIACT- 137 SMIACT-
50MHZIN 164 CLKIN
50MSRC 176 CLKSRC
14MHZ 186 14MHZ
TURBO 135 TURBO
2056PG 224 PWRGD
CACHE- 140 CACHE-
FERR- 138 FERR-
HITM- 139 HITM-
FROM FIG. 11A
CPU
I/F
A2 105 A2
A3 106 A3
A4 107 A4
A5 108 A5
A6 109 A6
A7 110 A7
A8 111 A8
A9 112 A9
A10 113 A10
A11 115 A11
A12 116 A12
A13 117 A13
A14 118 A14
A15 119 A15
A16 120 A16
A17 121 A17
A18 122 A18
A19 123 A19
A20 124 A20
A21 125 A21
A22 126 A22
A23 127 A23
A24 130 A24
A25 131 A25
A26 132 A26
A31_CS- 133 A31
GATEA20 134 GATEA20 [02]
SMI- 83 SMI- [02]
STPCLK- 84 STPCLK- [02]
CLKOUT 162 50MOUT [12]
CPURST 80 CPURST [02]
INTR 82 INTR [02]
NMI 86 NMI [02]
BRDY- 143 BRDY- [02]
EADS- 89 EADS- [02]
PHOLD 81 PHOLD [02]
NA- 142 2056NA- R309 NA- [02]
OPEN
IGNNE- 85 IGNNE- [02]
KEN- 87 KEN- [02]
FLUSH- 88 FLUSH- [02]
TO FIG. 11D
192 129
193 128
2056 PINOUT
256 65
1 64

FIG. 11B

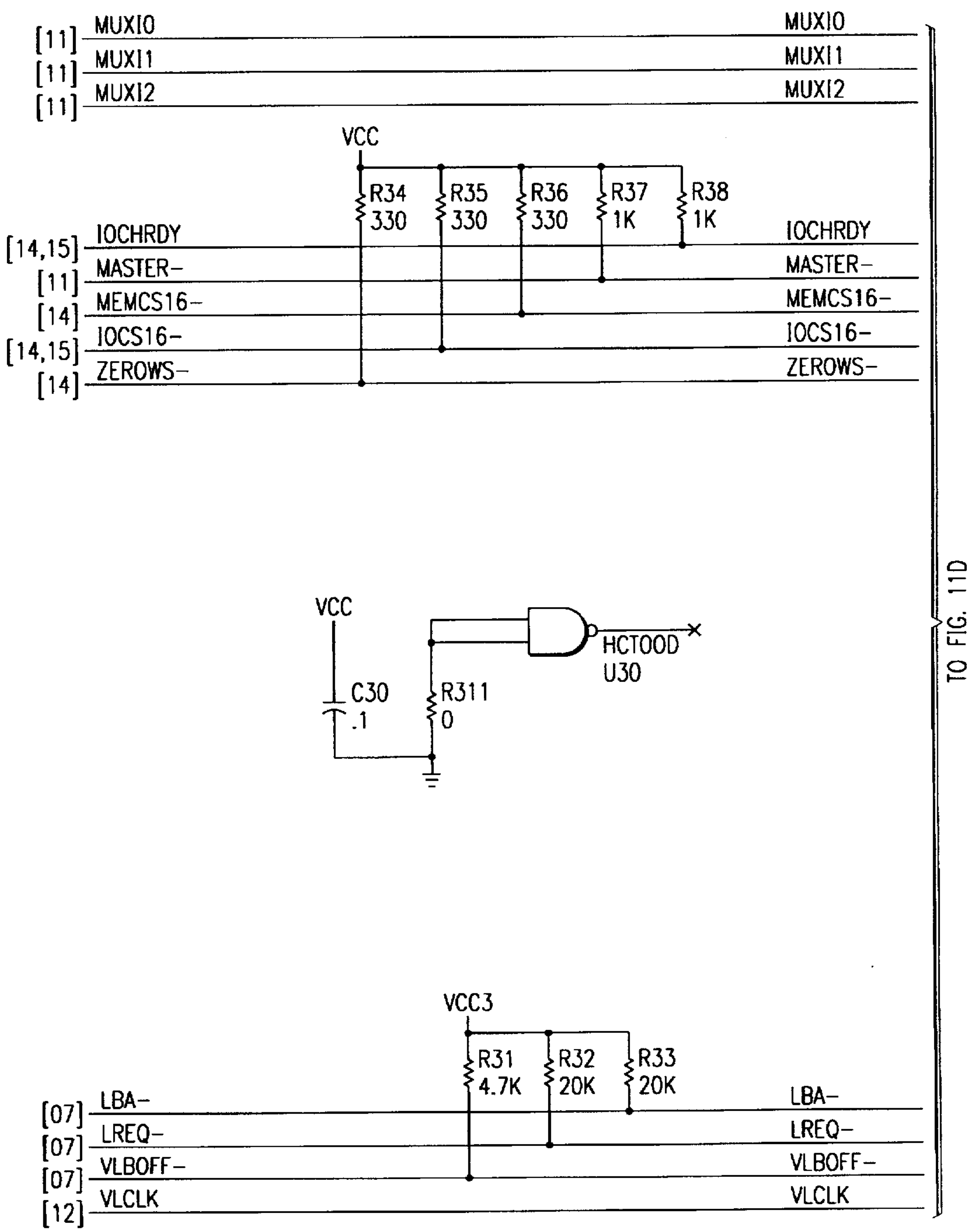

[11] MUXI0
[11] MUXI1
[11] MUXI2
MUXI0
MUXI1
MUXI2
VCC
R34 330
R35 330
R36 330
R37 1K
R38 1K
[14,15] IOCHRDY
[11] MASTER-
[14] MEMCS16-
[14,15] IOCS16-
[14] ZEROWS-
IOCHRDY
MASTER-
MEMCS16-
IOCS16-
ZEROWS-
VCC
C30 .1
R311 0
HCT00D U30
TO FIG. 11D
VCC3
R31 4.7K
R32 20K
R33 20K
[07] LBA-
[07] LREQ-
[07] VLBOFF-
[12] VLCLK
LBA-
LREQ-
VLBOFF-
VLCLK

FIG. 11C

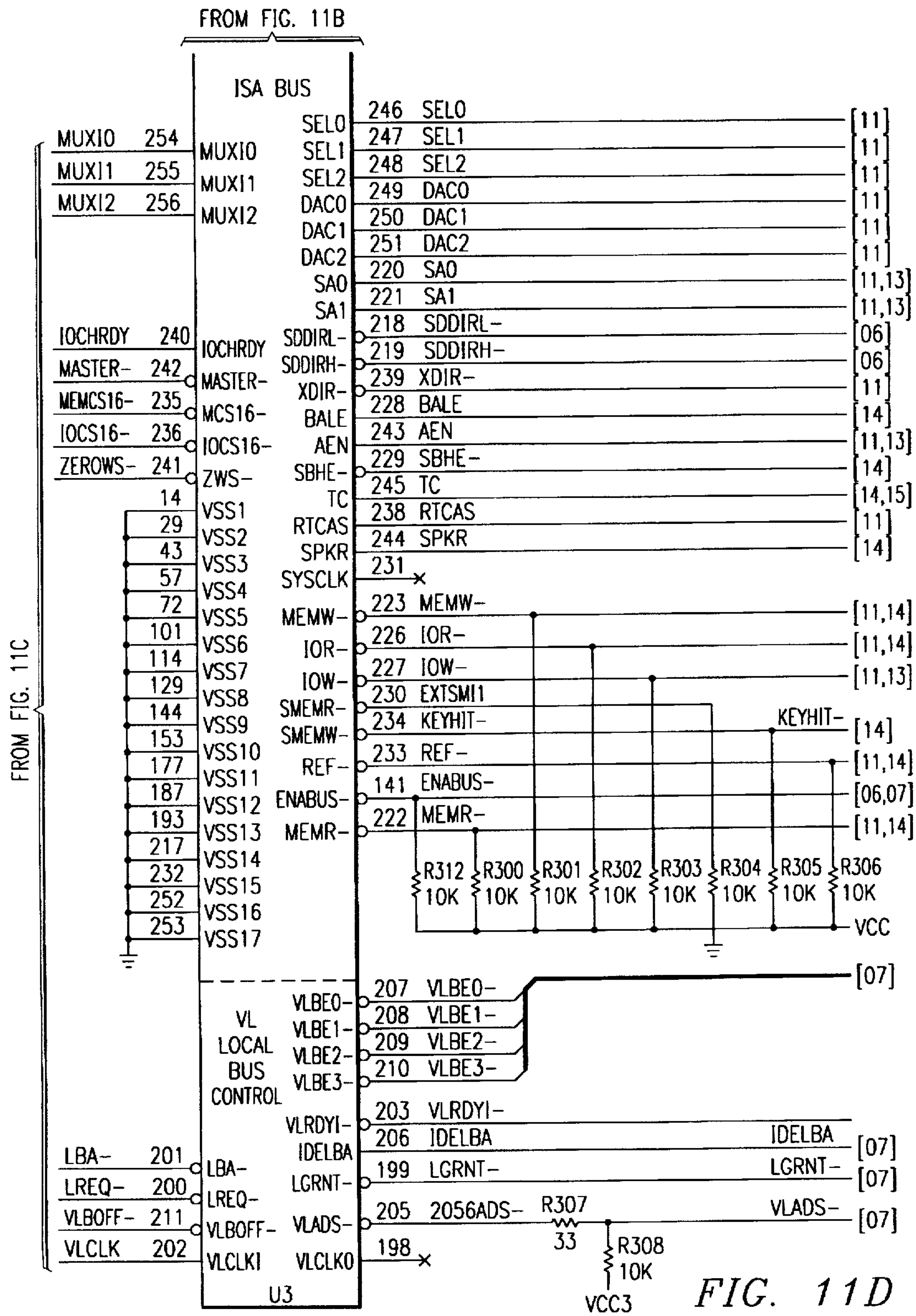
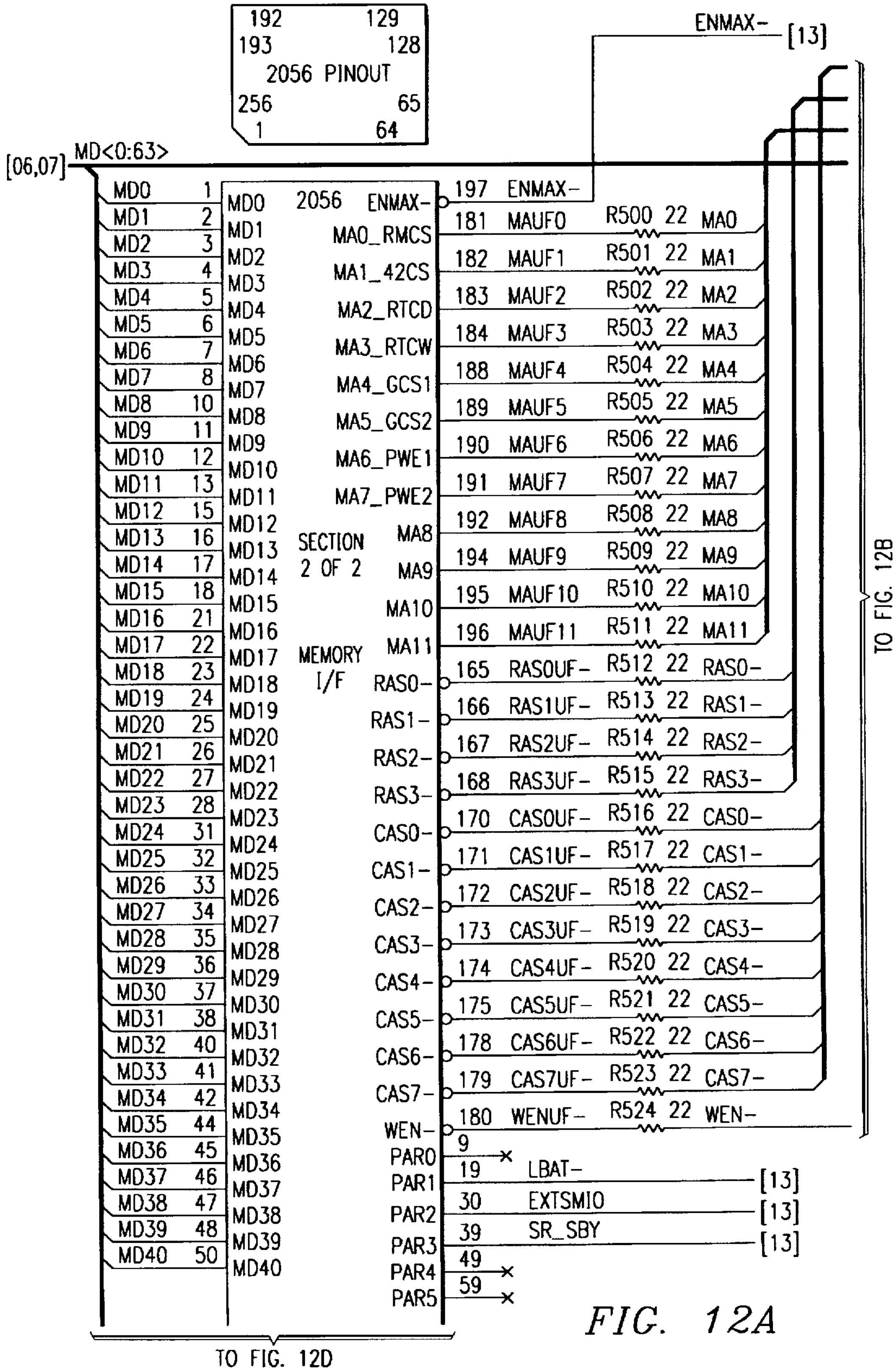

FROM FIG. 11B
ISA BUS
FROM FIG. 11C
MUXI0 254
MUXI1 255
MUXI2 256
IOCHRDY 240
MASTER- 242
MEMCS16- 235
IOCS16- 236
ZEROWS- 241
SEL0 SEL1 SEL2 DAC0 DAC1 DAC2 SA0 SA1 SDDIRL- SDDIRH- XDIR- BALE AEN SBHE- TC RTCAS SPKR SYSCLK MEMW- IOR- IOW- SMEMR- SMEMW- REF- ENABUS- MEMR-
246 SEL0 [11]
247 SEL1 [11]
248 SEL2 [11]
249 DAC0 [11]
250 DAC1 [11]
251 DAC2 [11]
220 SA0 [11,13]
221 SA1 [11,13]
218 SDDIRL- [06]
219 SDDIRH- [06]
239 XDIR- [11]
228 BALE [14]
243 AEN [11,13]
229 SBHE- [14]
245 TC [14,15]
238 RTCAS [11]
244 SPKR [14]
231
223 MEMW- [11,14]
226 IOR- [11,14]
227 IOW- [11,13]
230 EXTSMI1
234 KEYHIT- KEYHIT- [14]
233 REF- [11,14]
141 ENABUS- [06,07]
222 MEMR- [11,14]
R312 10K R300 10K R301 10K R302 10K R303 10K R304 10K R305 10K R306 10K
VCC
VSS1 ... VSS17: 14 29 43 57 72 101 114 129 144 153 177 187 193 217 232 252 253
VL LOCAL BUS CONTROL
207 VLBE0- [07]
208 VLBE1-
209 VLBE2-
210 VLBE3-
203 VLRDYI-
206 IDELBA IDELBA [07]
199 LGRNT- LGRNT- [07]
205 2056ADS- R307 33 VLADS- [07]
R308 10K VCC3
198
LBA- 201
LREQ- 200
VLBOFF- 211
VLCLK 202
LBA- LREQ- VLBOFF- VLCLKI VLRDYI- IDELBA LGRNT- VLADS- VLCLKO
U3

FIG. 11D



192 129
193 128
2056 PINOUT
256 65
1 64
ENMAX- [13]
[06,07] MD<0:63>
2056
SECTION 2 OF 2
MEMORY I/F
197 ENMAX-
181 MAUF0 R500 22 MA0
182 MAUF1 R501 22 MA1
183 MAUF2 R502 22 MA2
184 MAUF3 R503 22 MA3
188 MAUF4 R504 22 MA4
189 MAUF5 R505 22 MA5
190 MAUF6 R506 22 MA6
191 MAUF7 R507 22 MA7
192 MAUF8 R508 22 MA8
194 MAUF9 R509 22 MA9
195 MAUF10 R510 22 MA10
196 MAUF11 R511 22 MA11
165 RAS0UF- R512 22 RAS0-
166 RAS1UF- R513 22 RAS1-
167 RAS2UF- R514 22 RAS2-
168 RAS3UF- R515 22 RAS3-
170 CAS0UF- R516 22 CAS0-
171 CAS1UF- R517 22 CAS1-
172 CAS2UF- R518 22 CAS2-
173 CAS3UF- R519 22 CAS3-
174 CAS4UF- R520 22 CAS4-
175 CAS5UF- R521 22 CAS5-
178 CAS6UF- R522 22 CAS6-
179 CAS7UF- R523 22 CAS7-
180 WENUF- R524 22 WEN-
ENMAX- MA0_RMCS MA1_42CS MA2_RTCD MA3_RTCW MA4_GCS1 MA5_GCS2 MA6_PWE1 MA7_PWE2 MA8 MA9 MA10 MA11 RAS0- RAS1- RAS2- RAS3- CAS0- CAS1- CAS2- CAS3- CAS4- CAS5- CAS6- CAS7- WEN- PAR0 PAR1 PAR2 PAR3 PAR4 PAR5
9
19 LBAT- [13]
30 EXTSMI0 [13]
39 SR_SBY [13]
49
59
MD0 1 MD0
MD1 2 MD1
MD2 3 MD2
MD3 4 MD3
MD4 5 MD4
MD5 6 MD5
MD6 7 MD6
MD7 8 MD7
MD8 10 MD8
MD9 11 MD9
MD10 12 MD10
MD11 13 MD11
MD12 15 MD12
MD13 16 MD13
MD14 17 MD14
MD15 18 MD15
MD16 21 MD16
MD17 22 MD17
MD18 23 MD18
MD19 24 MD19
MD20 25 MD20
MD21 26 MD21
MD22 27 MD22
MD23 28 MD23
MD24 31 MD24
MD25 32 MD25
MD26 33 MD26
MD27 34 MD27
MD28 35 MD28
MD29 36 MD29
MD30 37 MD30
MD31 38 MD31
MD32 40 MD32
MD33 41 MD33
MD34 42 MD34
MD35 44 MD35
MD36 45 MD36
MD37 46 MD37
MD38 47 MD38
MD39 48 MD39
MD40 50 MD40
TO FIG. 12B
TO FIG. 12D

FIG. 12A

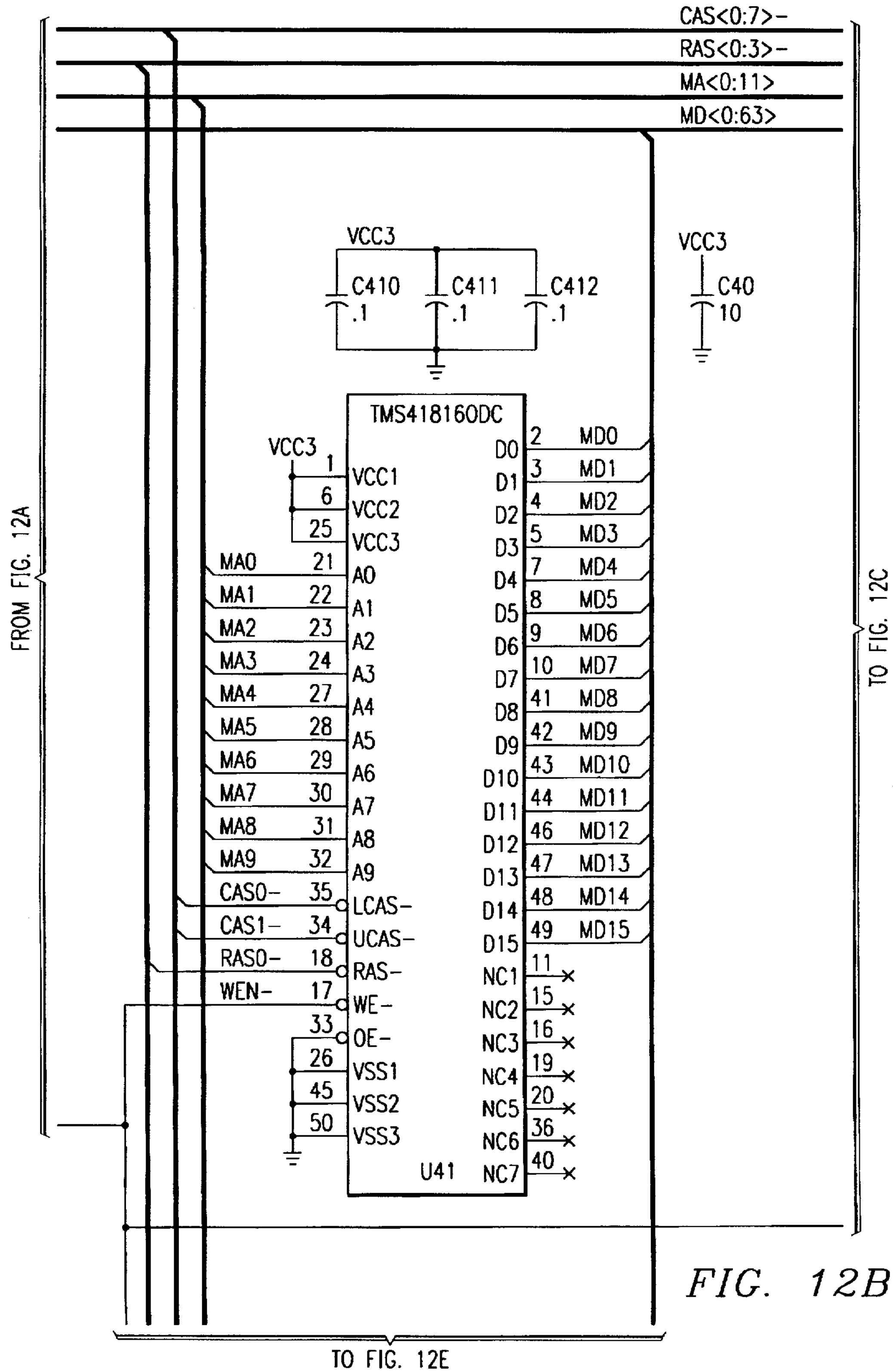
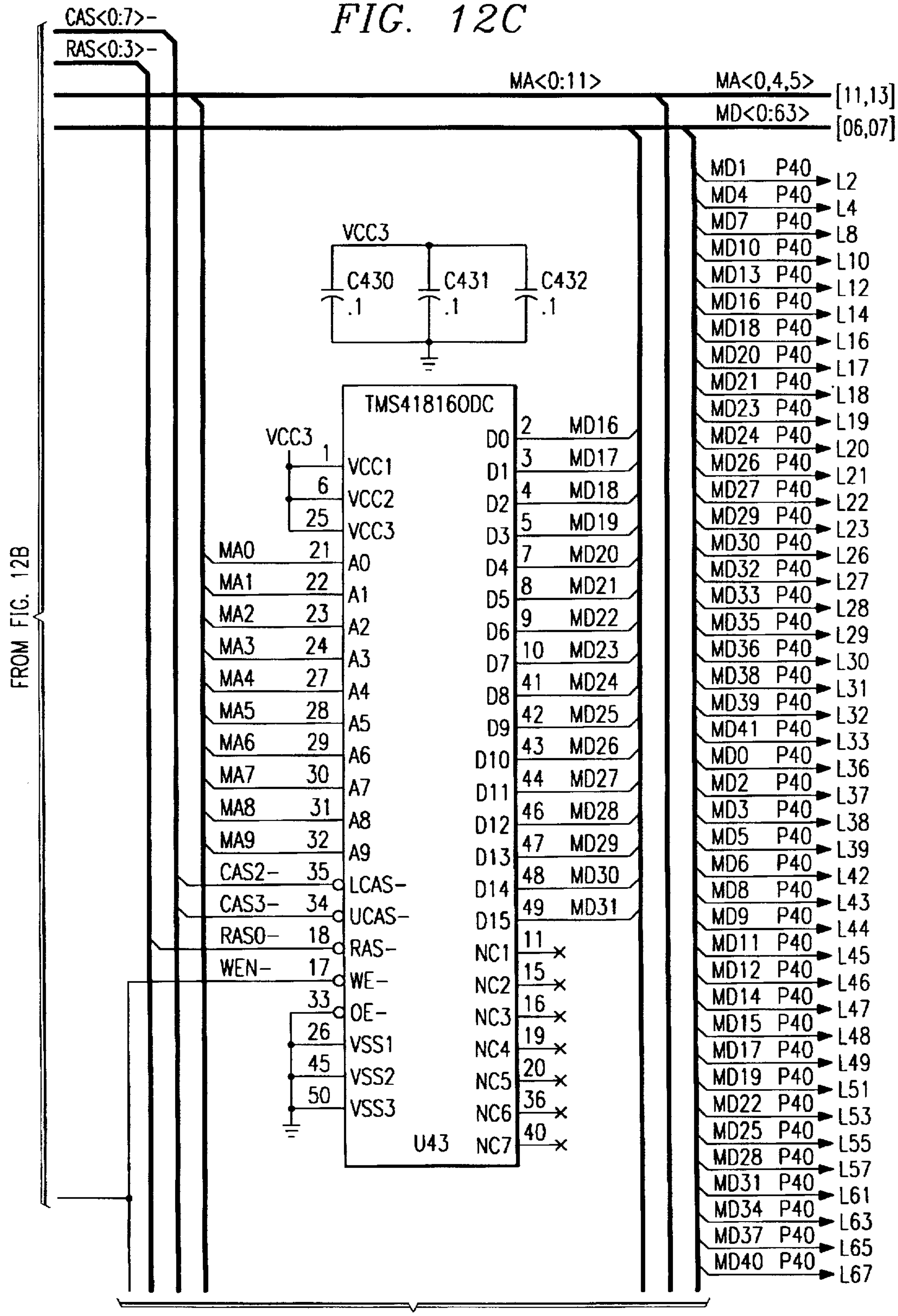

CAS<0:7>-
RAS<0:3>-
MA<0:11>
MD<0:63>
VCC3
C410
.1
C411
.1
C412
.1
VCC3
C40
10
TMS418160DC
VCC3
VCC1
VCC2
VCC3
MA0 21 A0
MA1 22 A1
MA2 23 A2
MA3 24 A3
MA4 27 A4
MA5 28 A5
MA6 29 A6
MA7 30 A7
MA8 31 A8
MA9 32 A9
CAS0- 35 LCAS-
CAS1- 34 UCAS-
RAS0- 18 RAS-
WEN- 17 WE-
33 OE-
26 VSS1
45 VSS2
50 VSS3
U41
D0 2 MD0
D1 3 MD1
D2 4 MD2
D3 5 MD3
D4 7 MD4
D5 8 MD5
D6 9 MD6
D7 10 MD7
D8 41 MD8
D9 42 MD9
D10 43 MD10
D11 44 MD11
D12 46 MD12
D13 47 MD13
D14 48 MD14
D15 49 MD15
NC1 11
NC2 15
NC3 16
NC4 19
NC5 20
NC6 36
NC7 40
FROM FIG. 12A
TO FIG. 12C
TO FIG. 12E

FIG. 12B

FIG. 12C



CAS<0:7>-
RAS<0:3>-
MA<0:11>
MA<0,4,5>
[11,13]
MD<0:63>
[06,07]
VCC3
C430
C431
C432
.1
TMS418160DC
U43
FROM FIG. 12B
TO FIG. 12F

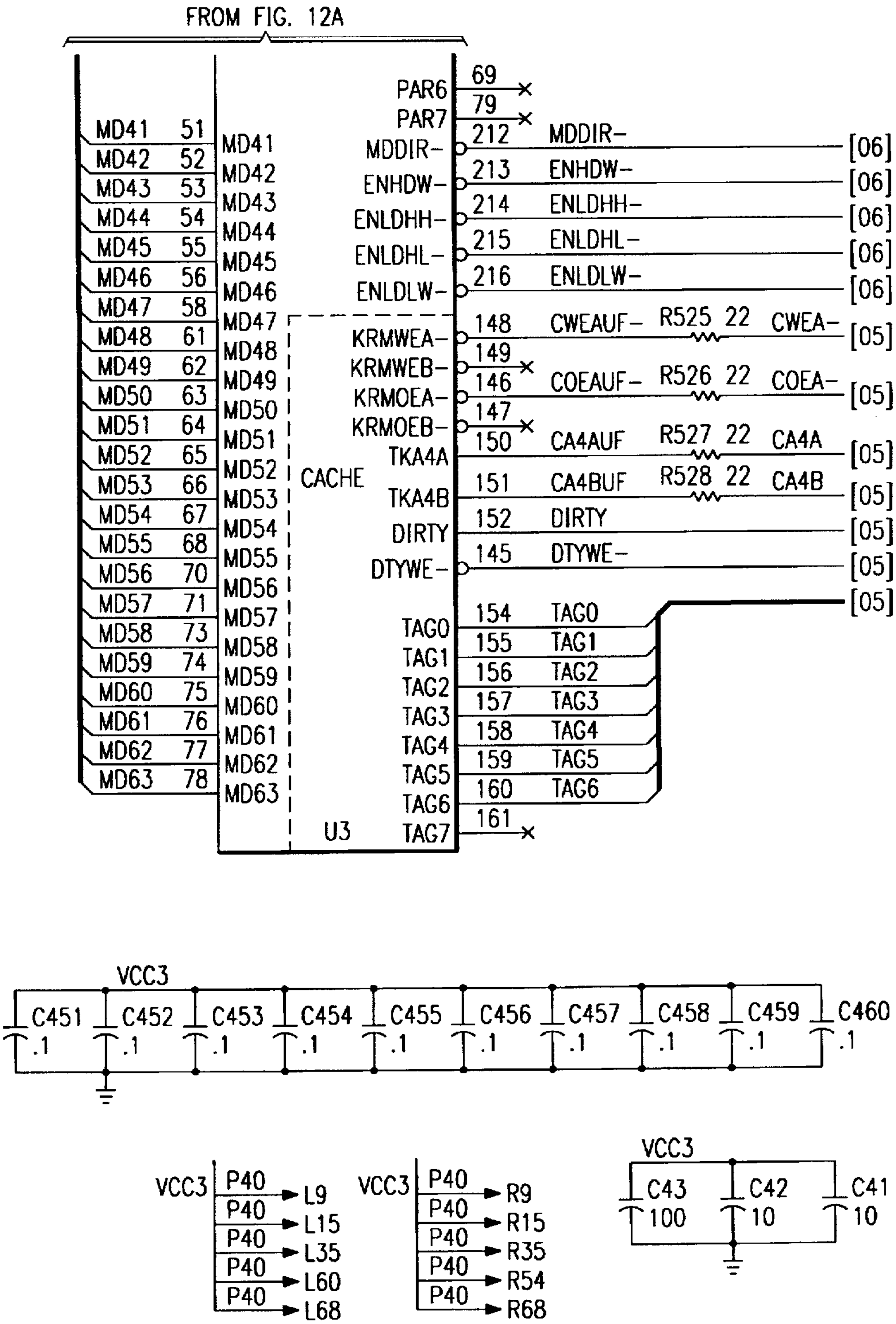
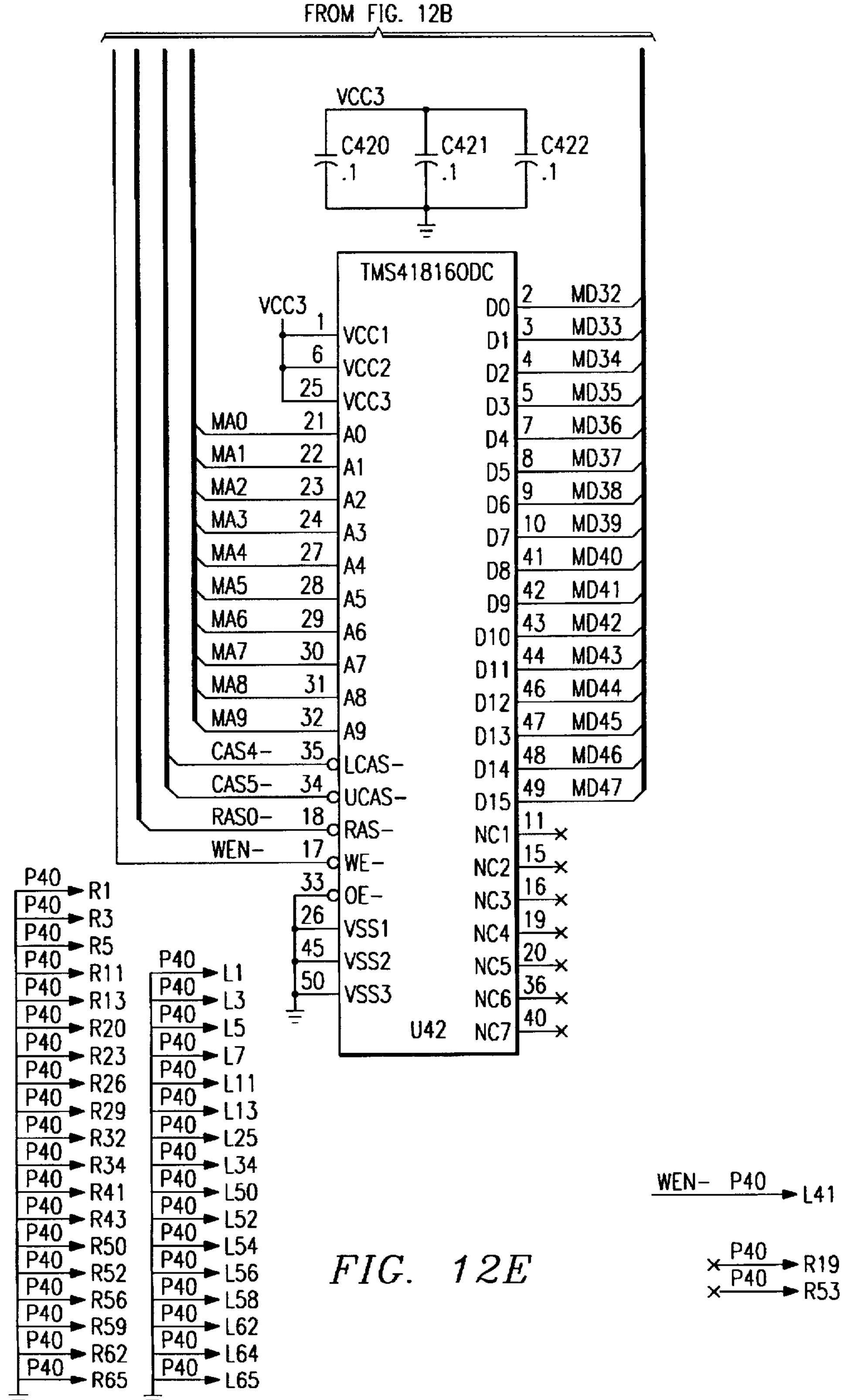

FROM FIG. 12A
MD41 51
MD42 52
MD43 53
MD44 54
MD45 55
MD46 56
MD47 58
MD48 61
MD49 62
MD50 63
MD51 64
MD52 65
MD53 66
MD54 67
MD55 68
MD56 70
MD57 71
MD58 73
MD59 74
MD60 75
MD61 76
MD62 77
MD63 78
CACHE
U3
PAR6 69
PAR7 79
MDDIR- 212 MDDIR- [06]
ENHDW- 213 ENHDW- [06]
ENLDHH- 214 ENLDHH- [06]
ENLDHL- 215 ENLDHL- [06]
ENLDLW- 216 ENLDLW- [06]
KRMWEA- 148 CWEAUF- R525 22 CWEA- [05]
KRMWEB- 149
KRMOEA- 146 COEAUF- R526 22 COEA- [05]
KRMOEB- 147
TKA4A 150 CA4AUF R527 22 CA4A [05]
TKA4B 151 CA4BUF R528 22 CA4B [05]
DIRTY 152 DIRTY [05]
DTYWE- 145 DTYWE- [05]
TAG0 154 TAG0 [05]
TAG1 155 TAG1
TAG2 156 TAG2
TAG3 157 TAG3
TAG4 158 TAG4
TAG5 159 TAG5
TAG6 160 TAG6
TAG7 161
VCC3
C451 .1
C452 .1
C453 .1
C454 .1
C455 .1
C456 .1
C457 .1
C458 .1
C459 .1
C460 .1
VCC3
P40 L9
P40 L15
P40 L35
P40 L60
P40 L68
VCC3
P40 R9
P40 R15
P40 R35
P40 R54
P40 R68
VCC3
C43 100
C42 10
C41 10

FIG. 12D



FROM FIG. 12B
VCC3
C420 .1
C421 .1
C422 .1
TMS418160DC
VCC3
1 VCC1
6 VCC2
25 VCC3
MA0 21 A0
MA1 22 A1
MA2 23 A2
MA3 24 A3
MA4 27 A4
MA5 28 A5
MA6 29 A6
MA7 30 A7
MA8 31 A8
MA9 32 A9
CAS4- 35 LCAS-
CAS5- 34 UCAS-
RAS0- 18 RAS-
WEN- 17 WE-
33 OE-
26 VSS1
45 VSS2
50 VSS3
U42
D0 2 MD32
D1 3 MD33
D2 4 MD34
D3 5 MD35
D4 7 MD36
D5 8 MD37
D6 9 MD38
D7 10 MD39
D8 41 MD40
D9 42 MD41
D10 43 MD42
D11 44 MD43
D12 46 MD44
D13 47 MD45
D14 48 MD46
D15 49 MD47
NC1 11
NC2 15
NC3 16
NC4 19
NC5 20
NC6 36
NC7 40
P40 R1
P40 R3
P40 R5
P40 R11
P40 R13
P40 R20
P40 R23
P40 R26
P40 R29
P40 R32
P40 R34
P40 R41
P40 R43
P40 R50
P40 R52
P40 R56
P40 R59
P40 R62
P40 R65
P40 L1
P40 L3
P40 L5
P40 L7
P40 L11
P40 L13
P40 L25
P40 L34
P40 L50
P40 L52
P40 L54
P40 L56
P40 L58
P40 L62
P40 L64
P40 L65
WEN- P40 L41
P40 R19
P40 R53

FIG. 12E

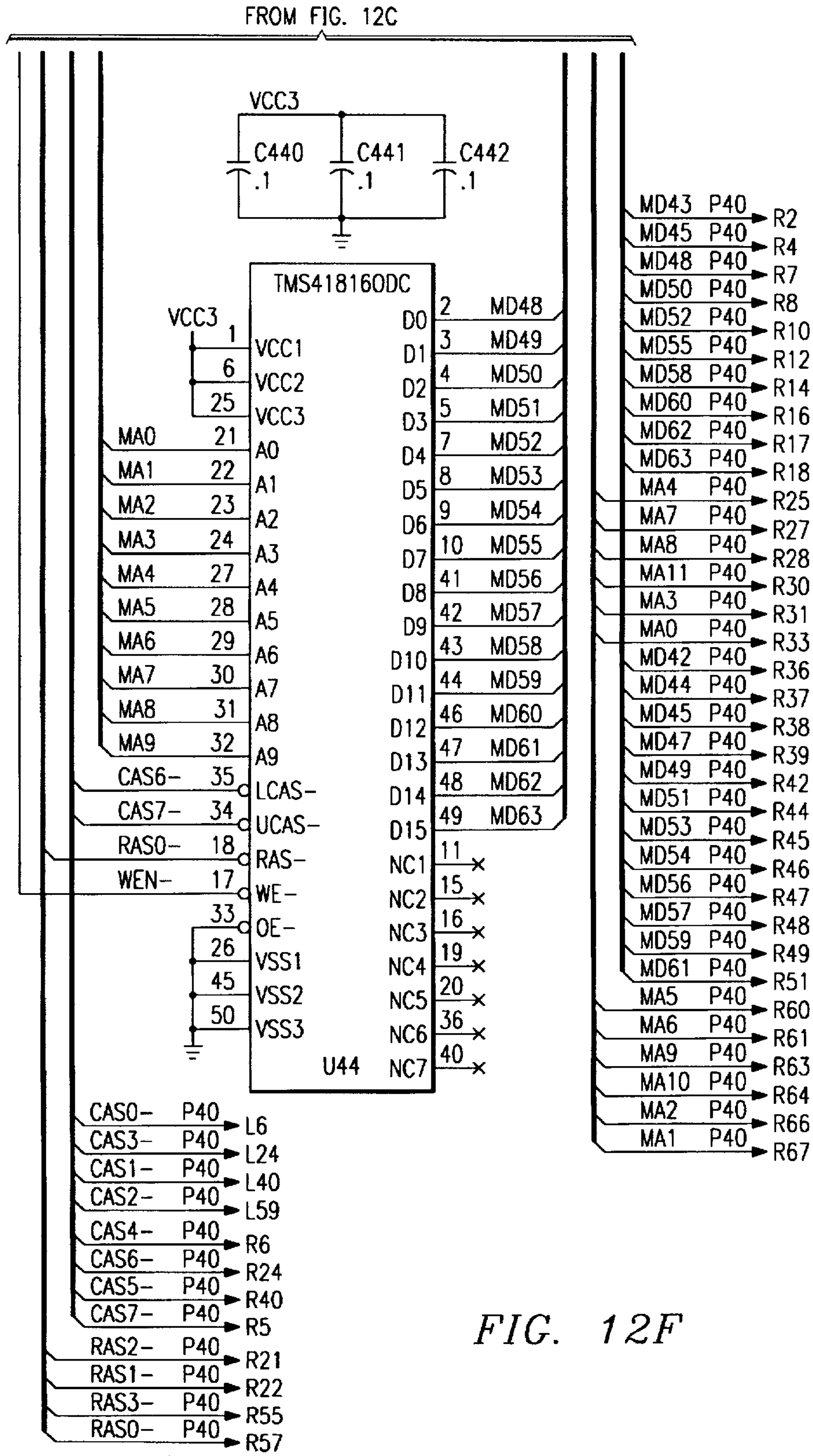

FROM FIG. 12C
VCC3
C440 .1
C441 .1
C442 .1
TMS418160DC
U44
VCC3
1 VCC1
6 VCC2
25 VCC3
MA0 21 A0
MA1 22 A1
MA2 23 A2
MA3 24 A3
MA4 27 A4
MA5 28 A5
MA6 29 A6
MA7 30 A7
MA8 31 A8
MA9 32 A9
CAS6- 35 LCAS-
CAS7- 34 UCAS-
RAS0- 18 RAS-
WEN- 17 WE-
33 OE-
26 VSS1
45 VSS2
50 VSS3
D0 2 MD48
D1 3 MD49
D2 4 MD50
D3 5 MD51
D4 7 MD52
D5 8 MD53
D6 9 MD54
D7 10 MD55
D8 41 MD56
D9 42 MD57
D10 43 MD58
D11 44 MD59
D12 46 MD60
D13 47 MD61
D14 48 MD62
D15 49 MD63
NC1 11
NC2 15
NC3 16
NC4 19
NC5 20
NC6 36
NC7 40
MD43 P40 R2
MD45 P40 R4
MD48 P40 R7
MD50 P40 R8
MD52 P40 R10
MD55 P40 R12
MD58 P40 R14
MD60 P40 R16
MD62 P40 R17
MD63 P40 R18
MA4 P40 R25
MA7 P40 R27
MA8 P40 R28
MA11 P40 R30
MA3 P40 R31
MA0 P40 R33
MD42 P40 R36
MD44 P40 R37
MD45 P40 R38
MD47 P40 R39
MD49 P40 R42
MD51 P40 R44
MD53 P40 R45
MD54 P40 R46
MD56 P40 R47
MD57 P40 R48
MD59 P40 R49
MD61 P40 R51
MA5 P40 R60
MA6 P40 R61
MA9 P40 R63
MA10 P40 R64
MA2 P40 R66
MA1 P40 R67
CAS0- P40 L6
CAS3- P40 L24
CAS1- P40 L40
CAS2- P40 L59
CAS4- P40 R6
CAS6- P40 R24
CAS5- P40 R40
CAS7- P40 R5
RAS2- P40 R21
RAS1- P40 R22
RAS3- P40 R55
RAS0- P40 R57

FIG. 12F

FIG. 13A

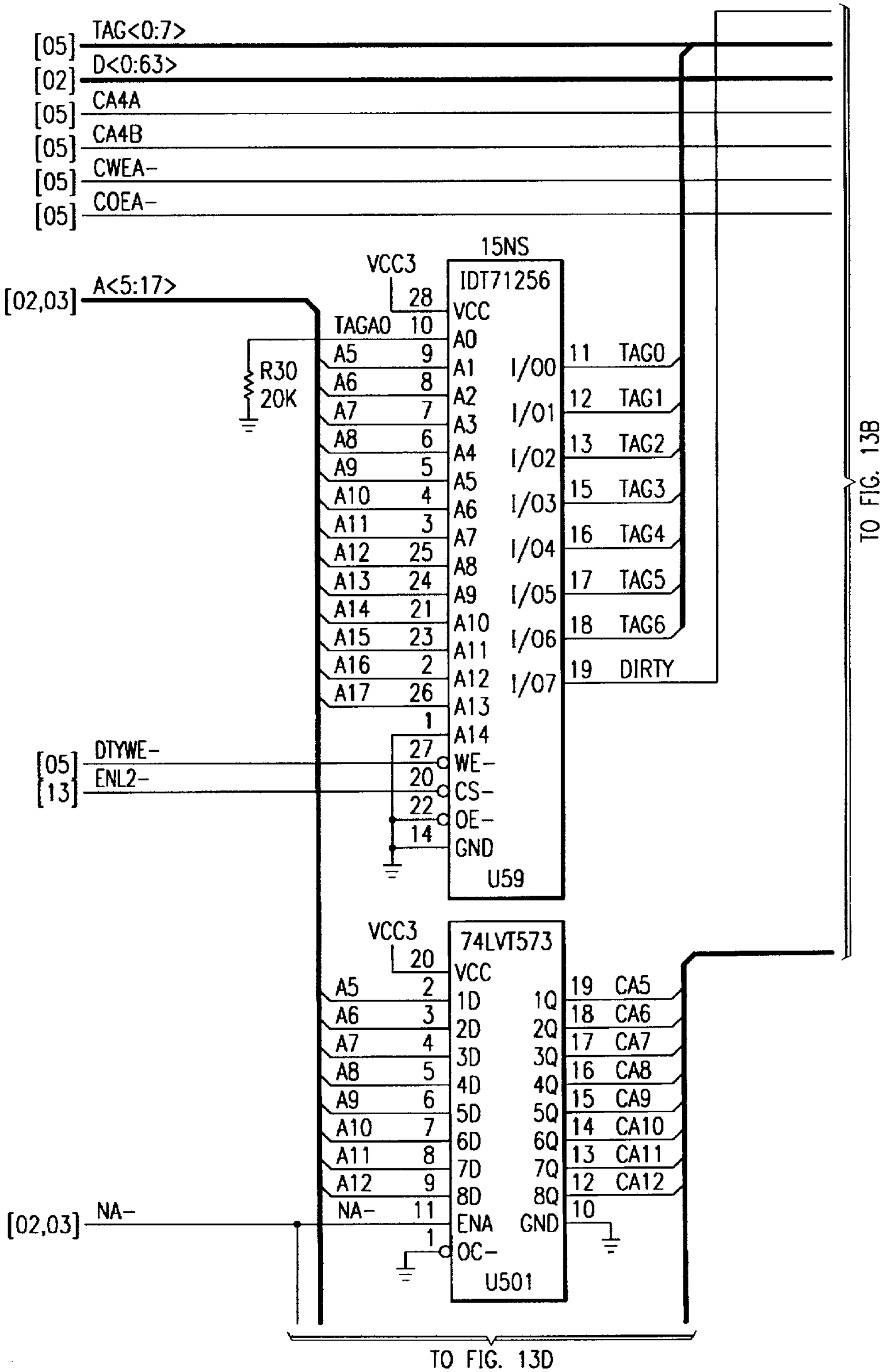

[05] TAG<0:7>
[02] D<0:63>
[05] CA4A
[05] CA4B
[05] CWEA-
[05] COEA-
[02,03] A<5:17>
VCC3
15NS
IDT71256
U59
R30
20K
TAGA0
DIRTY
[05] DTYWE-
[13] ENL2-
74LVT573
U501
[02,03] NA-
TO FIG. 13B
TO FIG. 13D

FIG. 13B

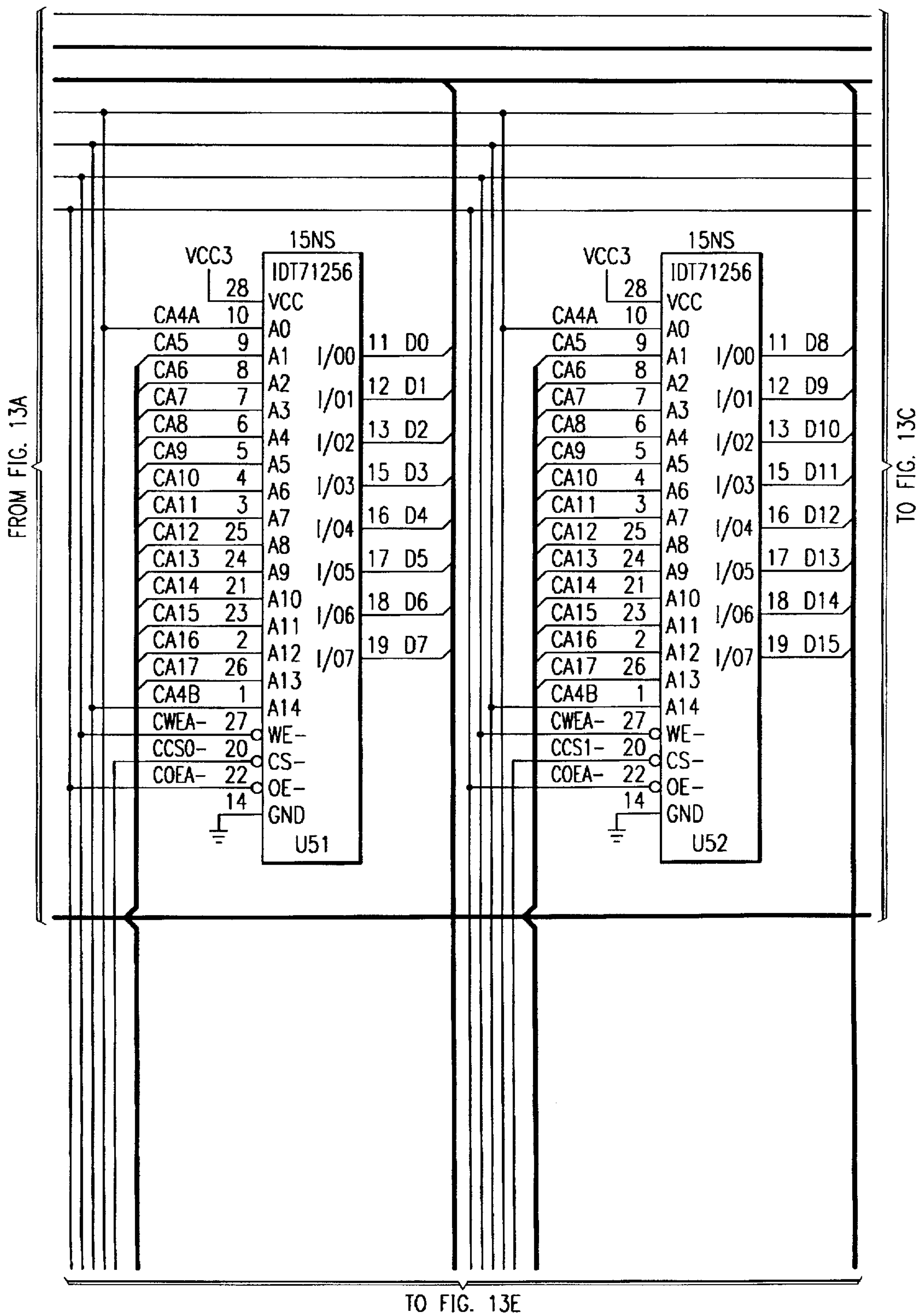

FROM FIG. 13A
TO FIG. 13C
TO FIG. 13E
15NS
IDT71256
VCC3
28 VCC
CA4A 10 A0
CA5 9 A1
CA6 8 A2
CA7 7 A3
CA8 6 A4
CA9 5 A5
CA10 4 A6
CA11 3 A7
CA12 25 A8
CA13 24 A9
CA14 21 A10
CA15 23 A11
CA16 2 A12
CA17 26 A13
CA4B 1 A14
CWEA- 27 WE-
CCS0- 20 CS-
COEA- 22 OE-
14 GND
U51
I/00 11 D0
I/01 12 D1
I/02 13 D2
I/03 15 D3
I/04 16 D4
I/05 17 D5
I/06 18 D6
I/07 19 D7
15NS
IDT71256
VCC3
28 VCC
CA4A 10 A0
CA5 9 A1
CA6 8 A2
CA7 7 A3
CA8 6 A4
CA9 5 A5
CA10 4 A6
CA11 3 A7
CA12 25 A8
CA13 24 A9
CA14 21 A10
CA15 23 A11
CA16 2 A12
CA17 26 A13
CA4B 1 A14
CWEA- 27 WE-
CCS1- 20 CS-
COEA- 22 OE-
14 GND
U52
I/00 11 D8
I/01 12 D9
I/02 13 D10
I/03 15 D11
I/04 16 D12
I/05 17 D13
I/06 18 D14
I/07 19 D15

FIG. 13C

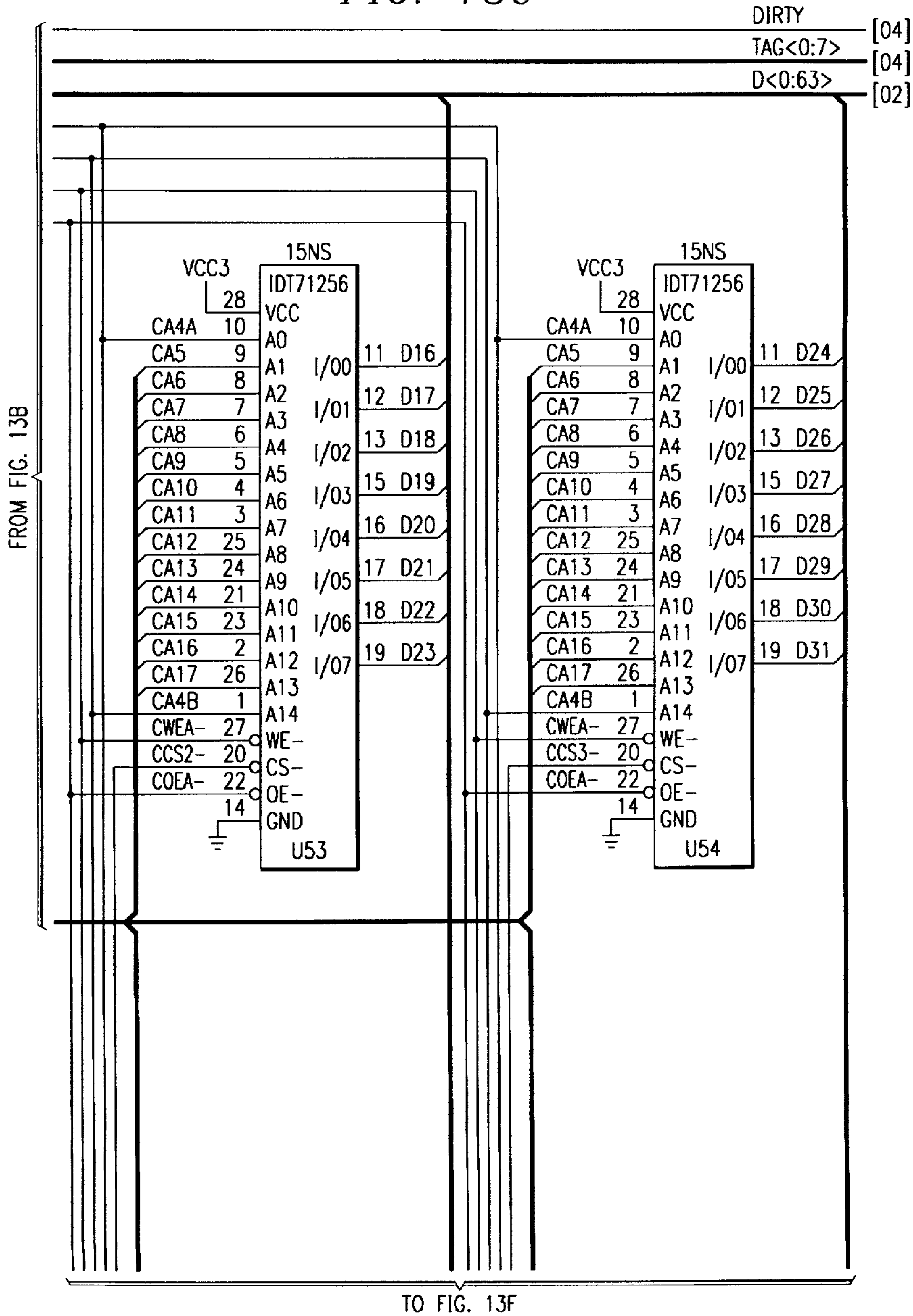
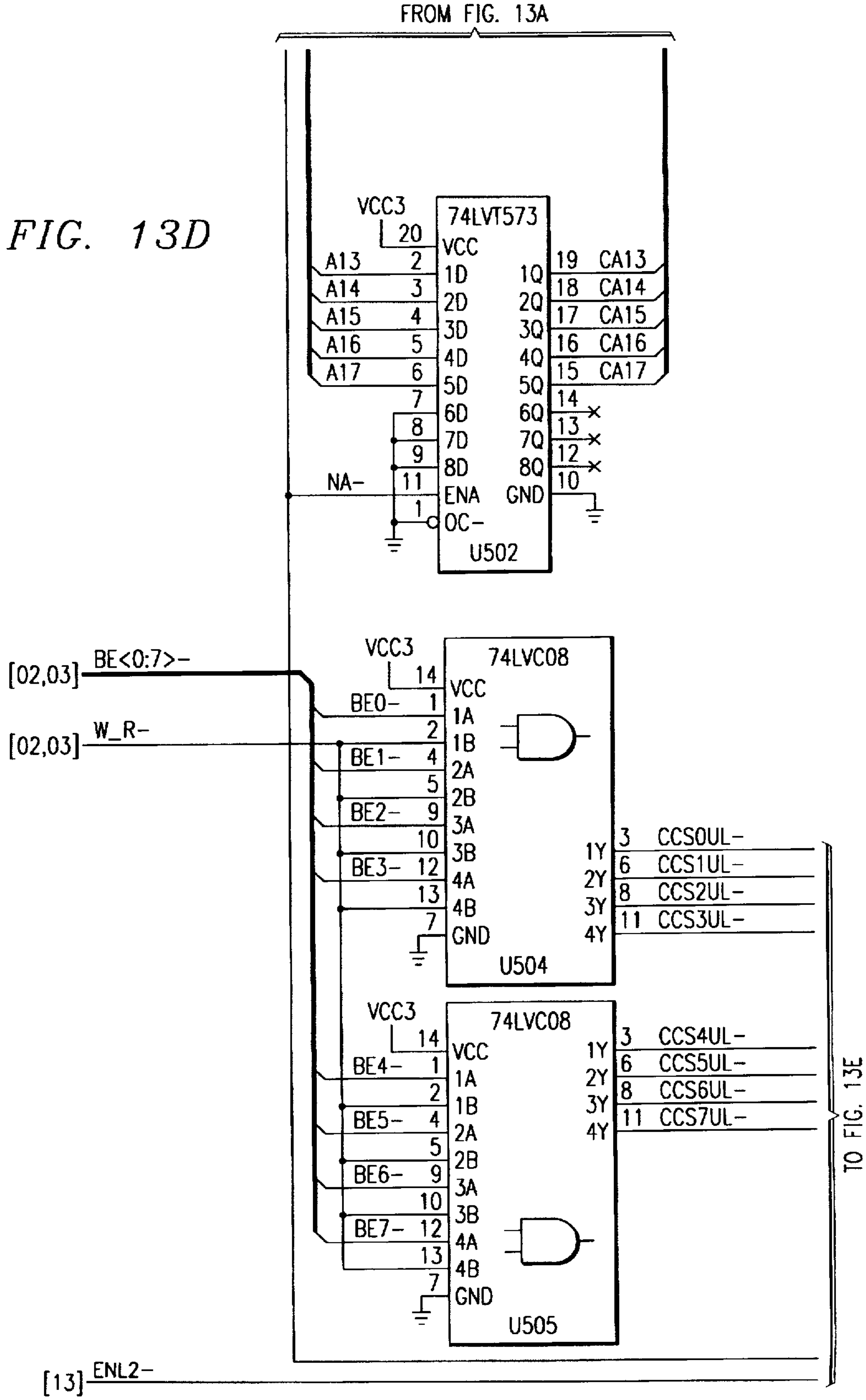
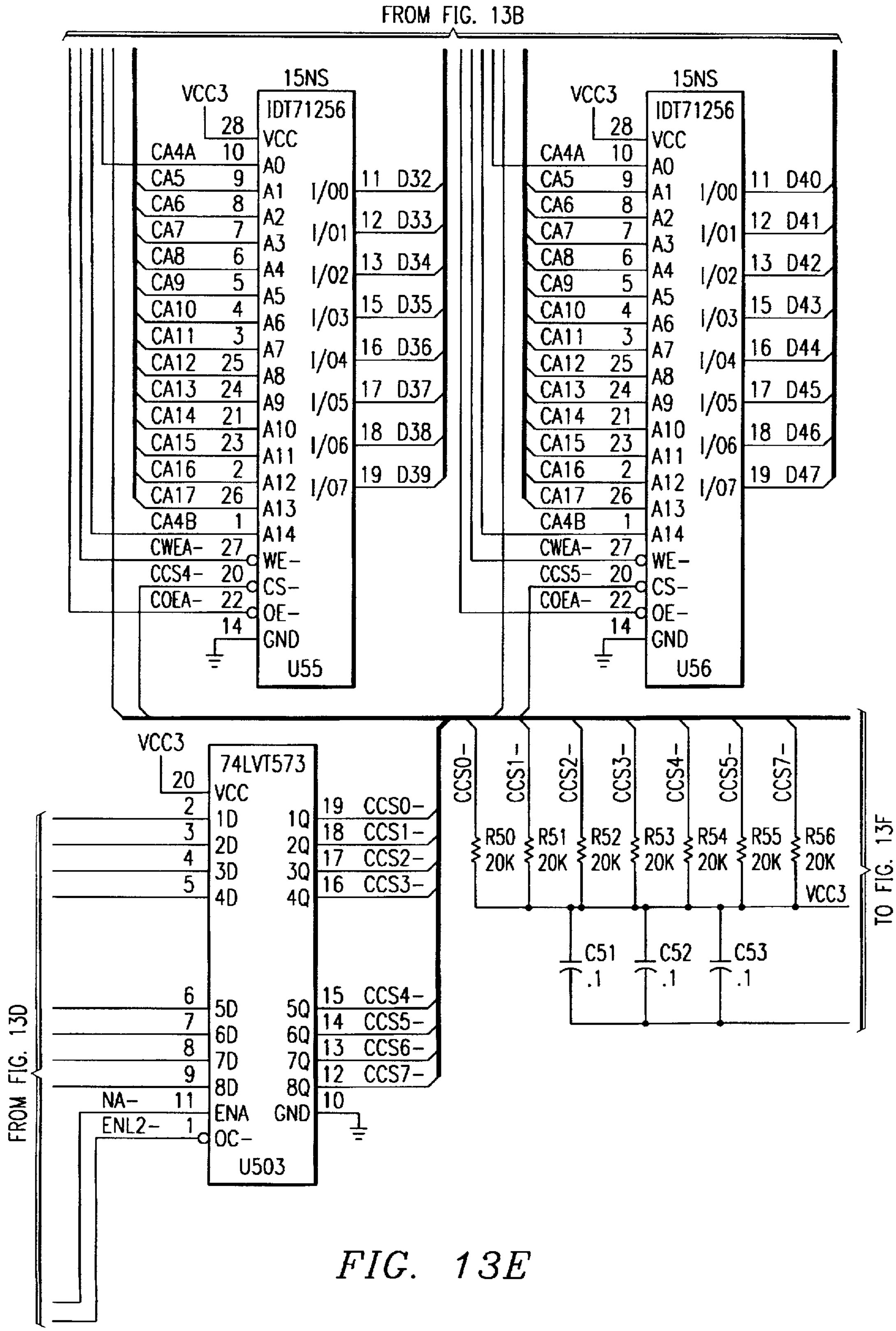

DIRTY
[04]
TAG<0:7>
[04]
D<0:63>
[02]
FROM FIG. 13B
15NS
VCC3
IDT71256
28 VCC
CA4A 10 A0
CA5 9 A1
CA6 8 A2
CA7 7 A3
CA8 6 A4
CA9 5 A5
CA10 4 A6
CA11 3 A7
CA12 25 A8
CA13 24 A9
CA14 21 A10
CA15 23 A11
CA16 2 A12
CA17 26 A13
CA4B 1 A14
CWEA- 27 WE-
CCS2- 20 CS-
COEA- 22 OE-
14 GND
U53
I/00 11 D16
I/01 12 D17
I/02 13 D18
I/03 15 D19
I/04 16 D20
I/05 17 D21
I/06 18 D22
I/07 19 D23
15NS
VCC3
IDT71256
CCS3- 20 CS-
U54
I/00 11 D24
I/01 12 D25
I/02 13 D26
I/03 15 D27
I/04 16 D28
I/05 17 D29
I/06 18 D30
I/07 19 D31
TO FIG. 13F

FIG. 13D


FROM FIG. 13A
VCC3
74LVT573
VCC
A13
A14
A15
A16
A17
1D
2D
3D
4D
5D
6D
7D
8D
ENA
OC-
1Q
2Q
3Q
4Q
5Q
6Q
7Q
8Q
GND
CA13
CA14
CA15
CA16
CA17
NA-
U502
[02,03]
BE<0:7>-
[02,03]
W_R-
VCC3
74LVC08
BE0-
BE1-
BE2-
BE3-
1A
1B
2A
2B
3A
3B
4A
4B
GND
1Y
2Y
3Y
4Y
CCS0UL-
CCS1UL-
CCS2UL-
CCS3UL-
U504
VCC3
74LVC08
BE4-
BE5-
BE6-
BE7-
CCS4UL-
CCS5UL-
CCS6UL-
CCS7UL-
U505
TO FIG. 13E
[13]
ENL2-



FROM FIG. 13B
15NS
IDT71256
VCC3
U55
U56
CA4A
CA5
CA6
CA7
CA8
CA9
CA10
CA11
CA12
CA13
CA14
CA15
CA16
CA17
CA4B
CWEA-
CCS4-
CCS5-
COEA-
VCC
GND
WE-
CS-
OE-
I/00
I/01
I/02
I/03
I/04
I/05
I/06
I/07
D32
D33
D34
D35
D36
D37
D38
D39
D40
D41
D42
D43
D44
D45
D46
D47
74LVT573
U503
1D
2D
3D
4D
5D
6D
7D
8D
1Q
2Q
3Q
4Q
5Q
6Q
7Q
8Q
ENA
OC-
NA-
ENL2-
CCS0-
CCS1-
CCS2-
CCS3-
CCS6-
CCS7-
R50
R51
R52
R53
R54
R55
R56
20K
C51
C52
C53
.1
TO FIG. 13F
FROM FIG. 13D

FIG. 13E

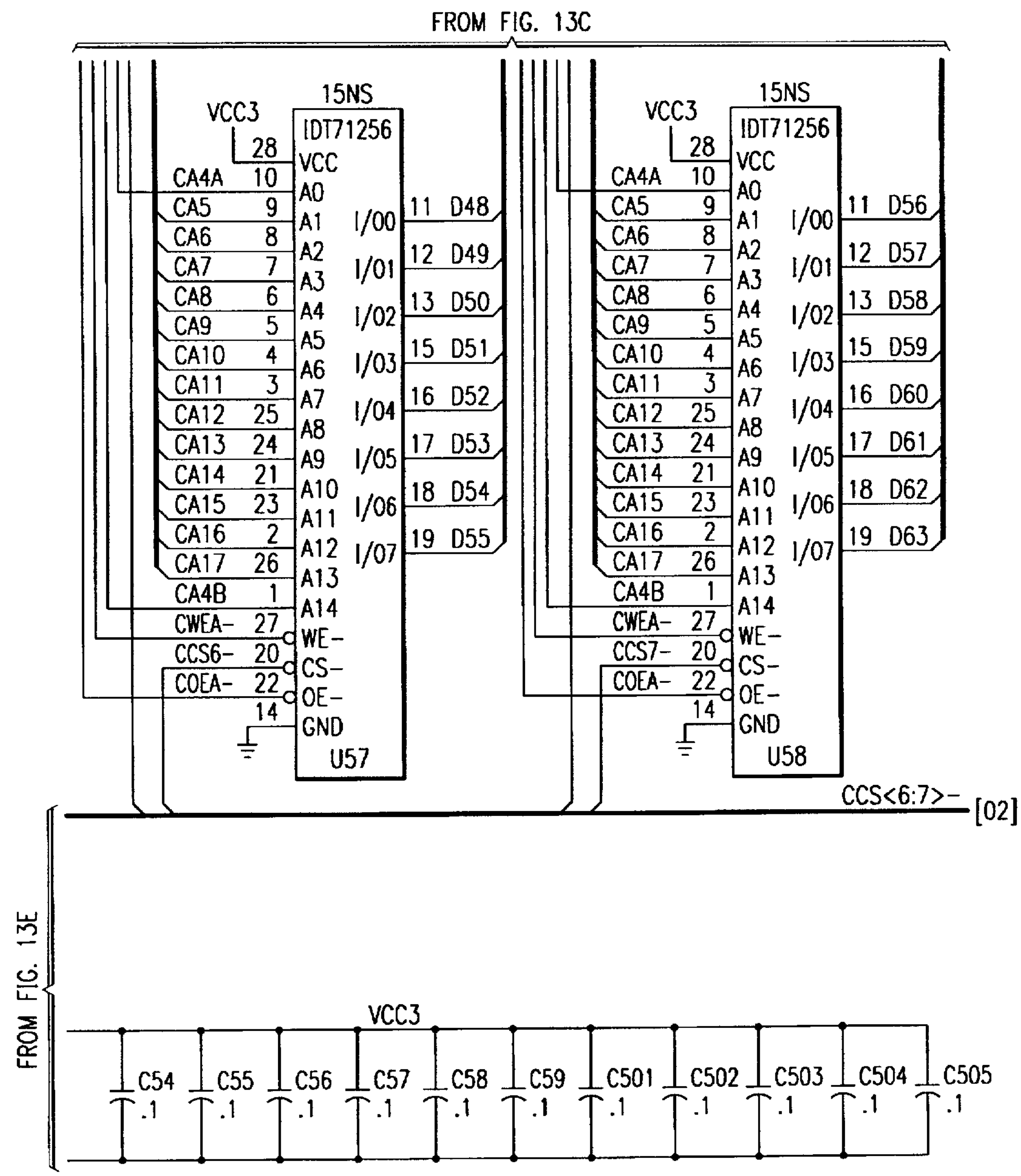
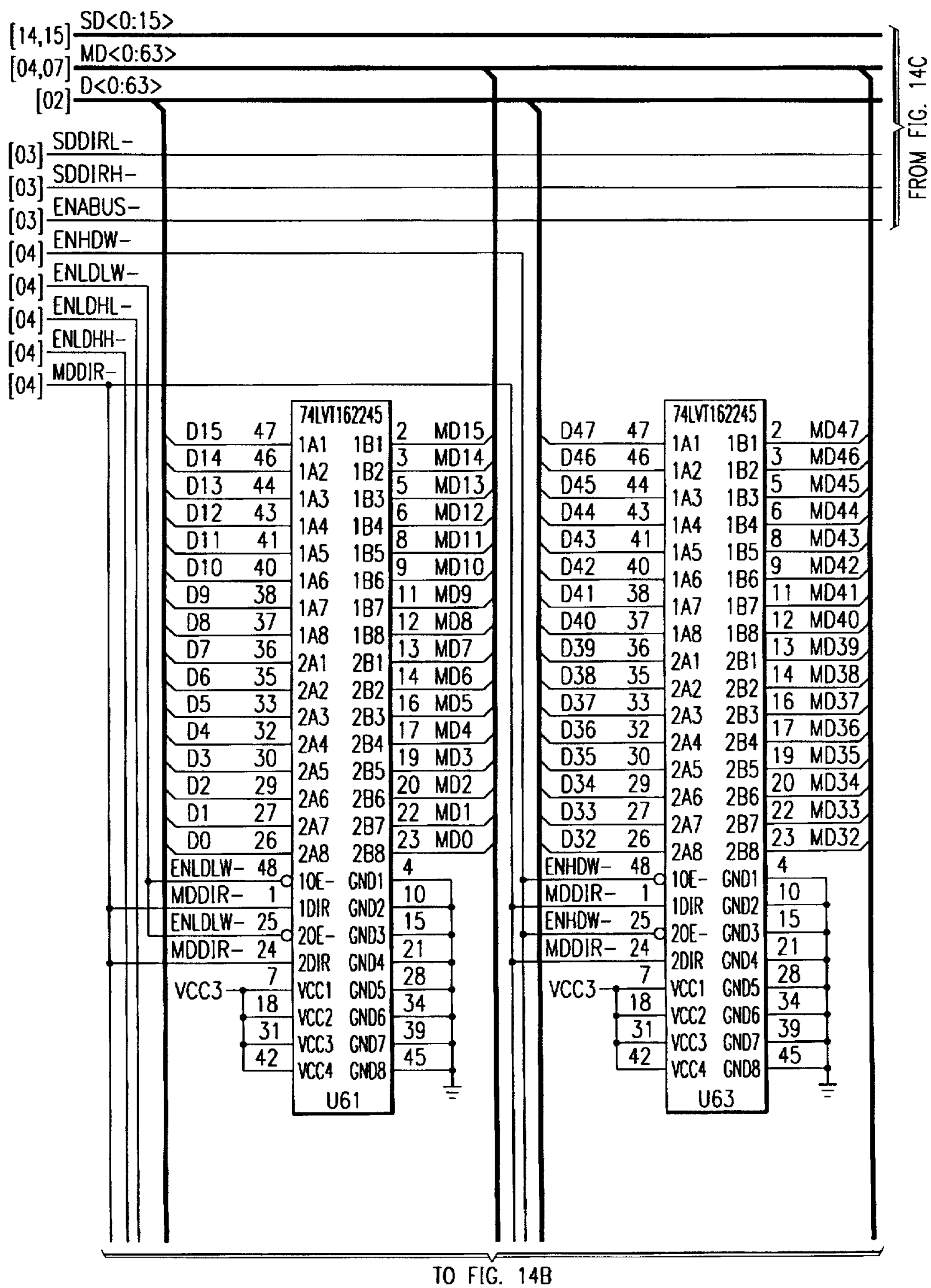

FROM FIG. 13C
15NS
IDT71256
VCC3
VCC
A0
A1
A2
A3
A4
A5
A6
A7
A8
A9
A10
A11
A12
A13
A14
WE-
CS-
OE-
GND
U57
CA4A 10
CA5 9
CA6 8
CA7 7
CA8 6
CA9 5
CA10 4
CA11 3
CA12 25
CA13 24
CA14 21
CA15 23
CA16 2
CA17 26
CA4B 1
CWEA- 27
CCS6- 20
COEA- 22
14
28
I/00 11 D48
I/01 12 D49
I/02 13 D50
I/03 15 D51
I/04 16 D52
I/05 17 D53
I/06 18 D54
I/07 19 D55
U58
CCS7- 20
I/00 11 D56
I/01 12 D57
I/02 13 D58
I/03 15 D59
I/04 16 D60
I/05 17 D61
I/06 18 D62
I/07 19 D63
CCS<6:7>- [02]
FROM FIG. 13E
VCC3
C54 .1
C55 .1
C56 .1
C57 .1
C58 .1
C59 .1
C501 .1
C502 .1
C503 .1
C504 .1
C505 .1

FIG. 13F

FIG. 14A



[14,15] SD<0:15>
[04,07] MD<0:63>
[02] D<0:63>
[03] SDDIRL-
[03] SDDIRH-
[03] ENABUS-
[04] ENHDW-
[04] ENLDLW-
[04] ENLDHL-
[04] ENLDHH-
[04] MDDIR-
FROM FIG. 14C
74LVT162245
U61
D15 47 1A1 1B1 2 MD15
D14 46 1A2 1B2 3 MD14
D13 44 1A3 1B3 5 MD13
D12 43 1A4 1B4 6 MD12
D11 41 1A5 1B5 8 MD11
D10 40 1A6 1B6 9 MD10
D9 38 1A7 1B7 11 MD9
D8 37 1A8 1B8 12 MD8
D7 36 2A1 2B1 13 MD7
D6 35 2A2 2B2 14 MD6
D5 33 2A3 2B3 16 MD5
D4 32 2A4 2B4 17 MD4
D3 30 2A5 2B5 19 MD3
D2 29 2A6 2B6 20 MD2
D1 27 2A7 2B7 22 MD1
D0 26 2A8 2B8 23 MD0
ENLDLW- 48 1OE- GND1 4
MDDIR- 1 1DIR GND2 10
ENLDLW- 25 2OE- GND3 15
MDDIR- 24 2DIR GND4 21
VCC3 7 VCC1 GND5 28
18 VCC2 GND6 34
31 VCC3 GND7 39
42 VCC4 GND8 45
74LVT162245
U63
D47 47 1A1 1B1 2 MD47
D46 46 1A2 1B2 3 MD46
D45 44 1A3 1B3 5 MD45
D44 43 1A4 1B4 6 MD44
D43 41 1A5 1B5 8 MD43
D42 40 1A6 1B6 9 MD42
D41 38 1A7 1B7 11 MD41
D40 37 1A8 1B8 12 MD40
D39 36 2A1 2B1 13 MD39
D38 35 2A2 2B2 14 MD38
D37 33 2A3 2B3 16 MD37
D36 32 2A4 2B4 17 MD36
D35 30 2A5 2B5 19 MD35
D34 29 2A6 2B6 20 MD34
D33 27 2A7 2B7 22 MD33
D32 26 2A8 2B8 23 MD32
ENHDW- 48 1OE- GND1 4
MDDIR- 1 1DIR GND2 10
ENHDW- 25 2OE- GND3 15
MDDIR- 24 2DIR GND4 21
VCC3 7 VCC1 GND5 28
18 VCC2 GND6 34
31 VCC3 GND7 39
42 VCC4 GND8 45
TO FIG. 14B

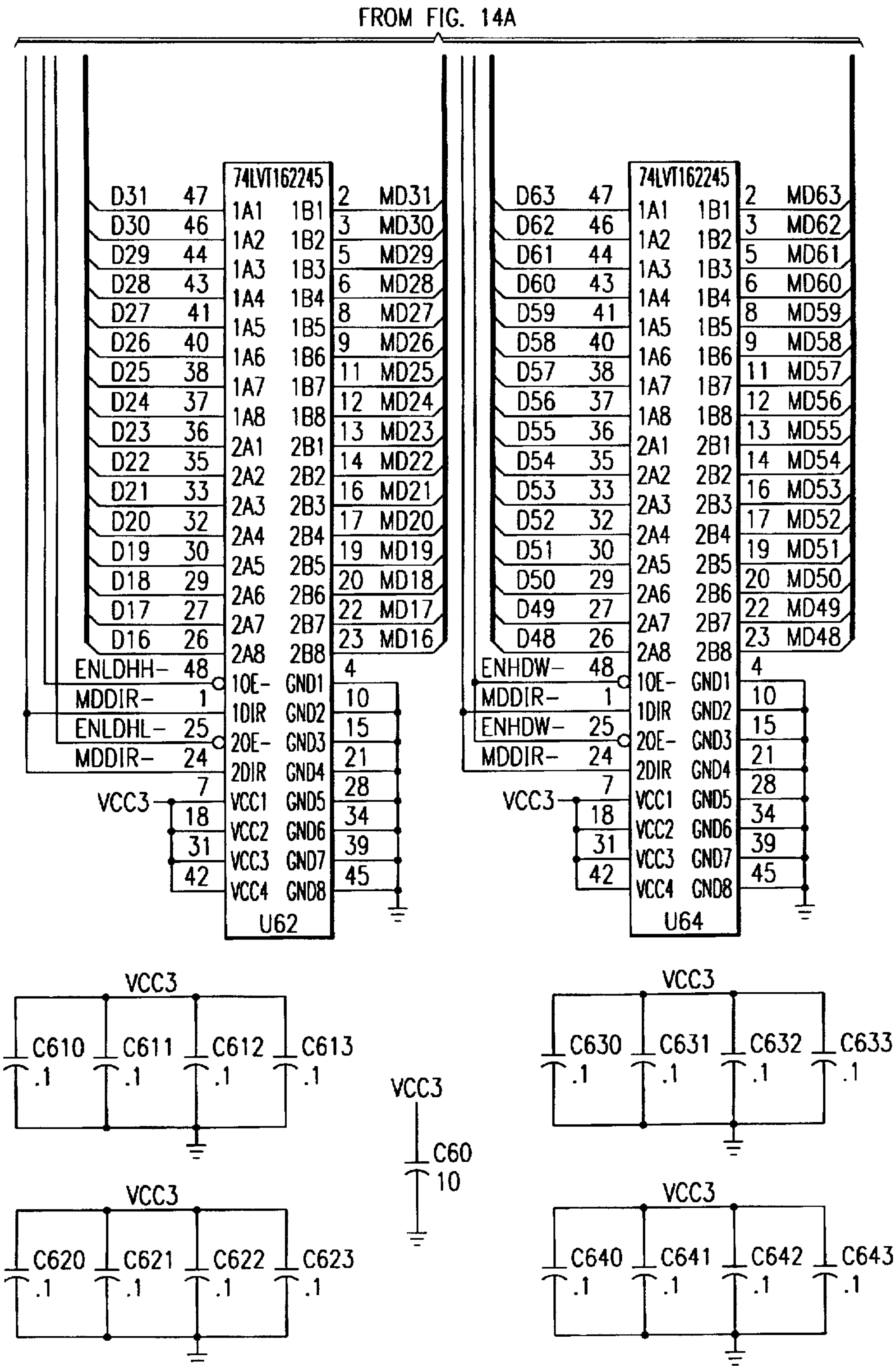

FROM FIG. 14A
74LVT162245
U62
U64
D31
MD31
D63
MD63
ENLDHH-
MDDIR-
ENLDHL-
ENHDW-
VCC3
1OE-
1DIR
2OE-
2DIR
VCC1
VCC2
VCC3
VCC4
GND1
GND8
C610
C611
C612
C613
C620
C621
C622
C623
C630
C631
C632
C633
C640
C641
C642
C643
C60
10
.1

FIG. 14B

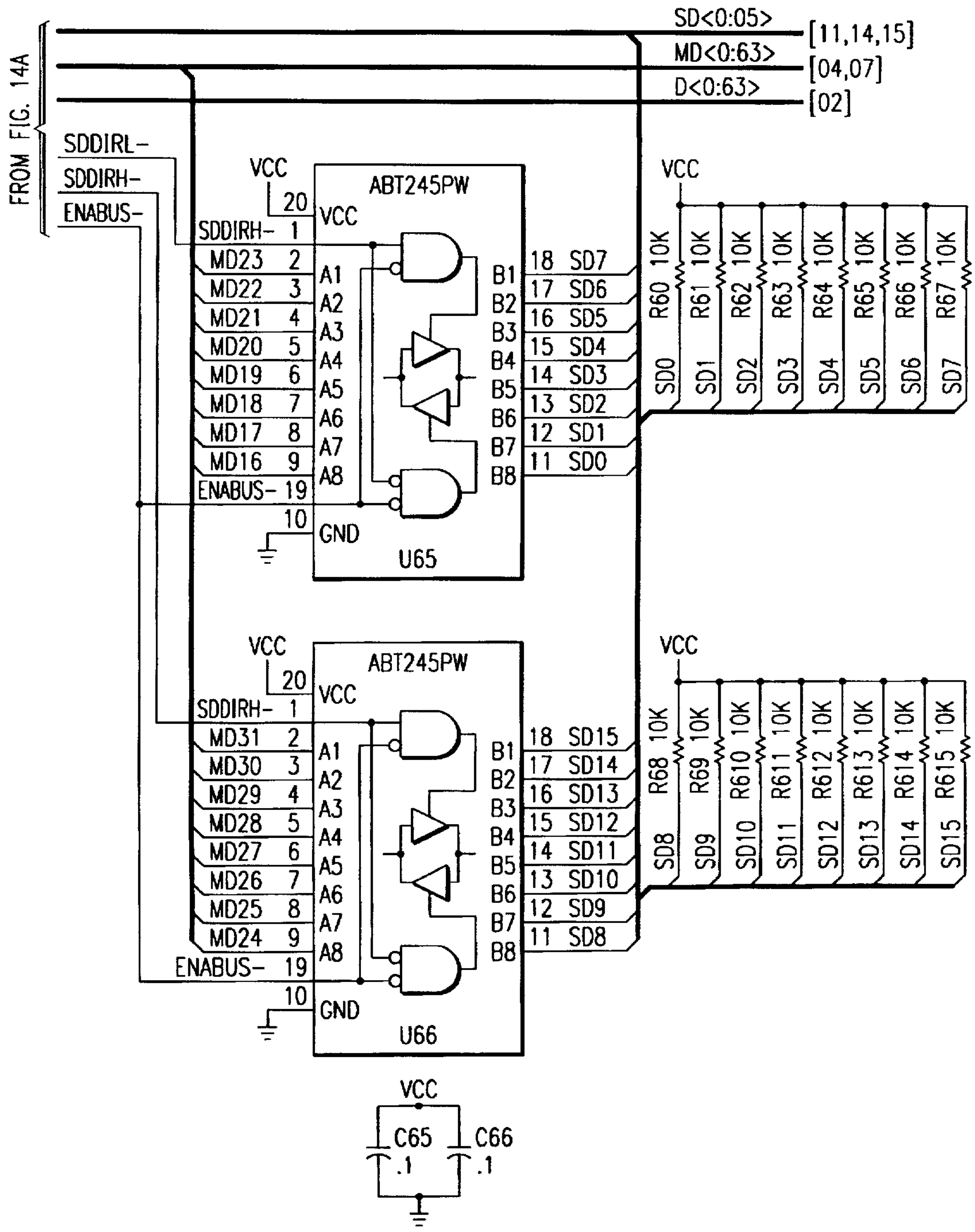
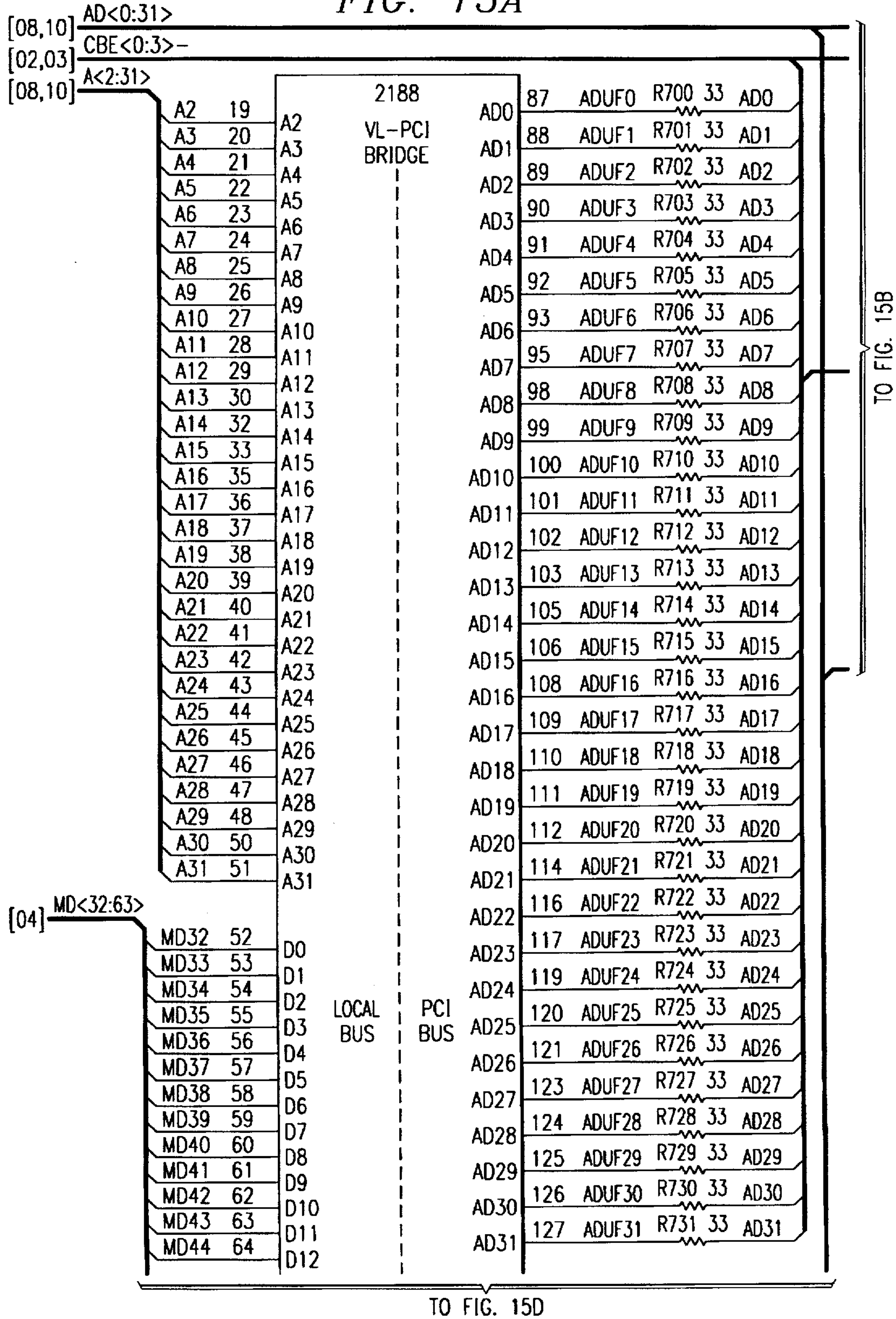

SD<0:05>
[11,14,15]
MD<0:63>
[04,07]
D<0:63>
[02]
FROM FIG. 14A
SDDIRL-
SDDIRH-
ENABUS-
VCC
ABT245PW
U65
U66
GND
VCC
C65
C66
.1

FIG. 14C

FIG. 15A


[08,10] AD<0:31>
[02,03] CBE<0:3>-
[08,10] A<2:31>
2188
VL-PCI
BRIDGE
A2 19 A2
A3 20 A3
A4 21 A4
A5 22 A5
A6 23 A6
A7 24 A7
A8 25 A8
A9 26 A9
A10 27 A10
A11 28 A11
A12 29 A12
A13 30 A13
A14 32 A14
A15 33 A15
A16 35 A16
A17 36 A17
A18 37 A18
A19 38 A19
A20 39 A20
A21 40 A21
A22 41 A22
A23 42 A23
A24 43 A24
A25 44 A25
A26 45 A26
A27 46 A27
A28 47 A28
A29 48 A29
A30 50 A30
A31 51 A31
[04] MD<32:63>
MD32 52 D0
MD33 53 D1
MD34 54 D2
MD35 55 D3
MD36 56 D4
MD37 57 D5
MD38 58 D6
MD39 59 D7
MD40 60 D8
MD41 61 D9
MD42 62 D10
MD43 63 D11
MD44 64 D12
LOCAL BUS
PCI BUS
AD0 87 ADUF0 R700 33 AD0
AD1 88 ADUF1 R701 33 AD1
AD2 89 ADUF2 R702 33 AD2
AD3 90 ADUF3 R703 33 AD3
AD4 91 ADUF4 R704 33 AD4
AD5 92 ADUF5 R705 33 AD5
AD6 93 ADUF6 R706 33 AD6
AD7 95 ADUF7 R707 33 AD7
AD8 98 ADUF8 R708 33 AD8
AD9 99 ADUF9 R709 33 AD9
AD10 100 ADUF10 R710 33 AD10
AD11 101 ADUF11 R711 33 AD11
AD12 102 ADUF12 R712 33 AD12
AD13 103 ADUF13 R713 33 AD13
AD14 105 ADUF14 R714 33 AD14
AD15 106 ADUF15 R715 33 AD15
AD16 108 ADUF16 R716 33 AD16
AD17 109 ADUF17 R717 33 AD17
AD18 110 ADUF18 R718 33 AD18
AD19 111 ADUF19 R719 33 AD19
AD20 112 ADUF20 R720 33 AD20
AD21 114 ADUF21 R721 33 AD21
AD22 116 ADUF22 R722 33 AD22
AD23 117 ADUF23 R723 33 AD23
AD24 119 ADUF24 R724 33 AD24
AD25 120 ADUF25 R725 33 AD25
AD26 121 ADUF26 R726 33 AD26
AD27 123 ADUF27 R727 33 AD27
AD28 124 ADUF28 R728 33 AD28
AD29 125 ADUF29 R729 33 AD29
AD30 126 ADUF30 R730 33 AD30
AD31 127 ADUF31 R731 33 AD31
TO FIG. 15B
TO FIG. 15D

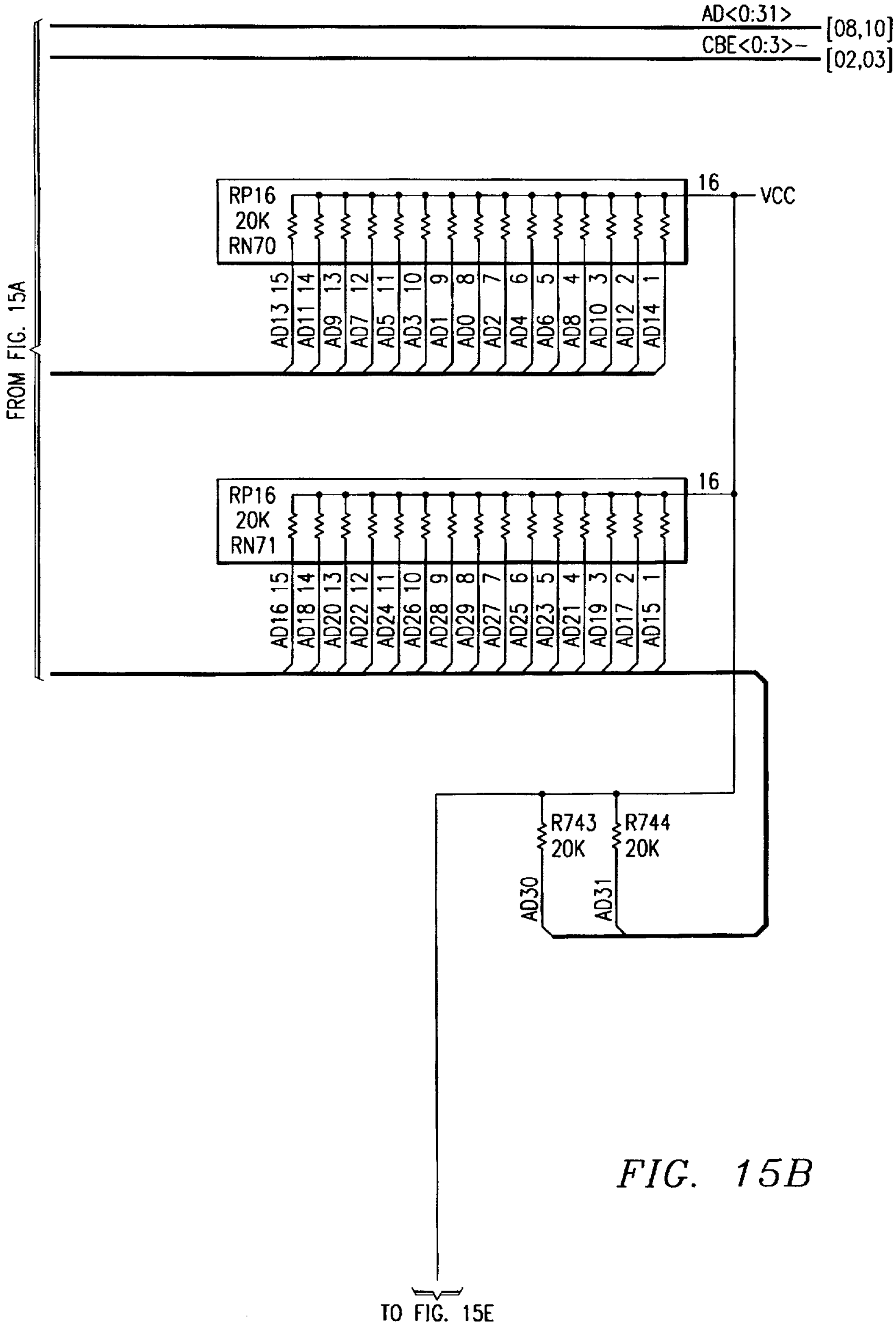
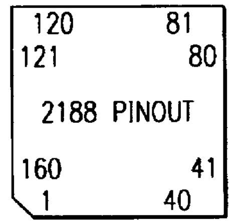
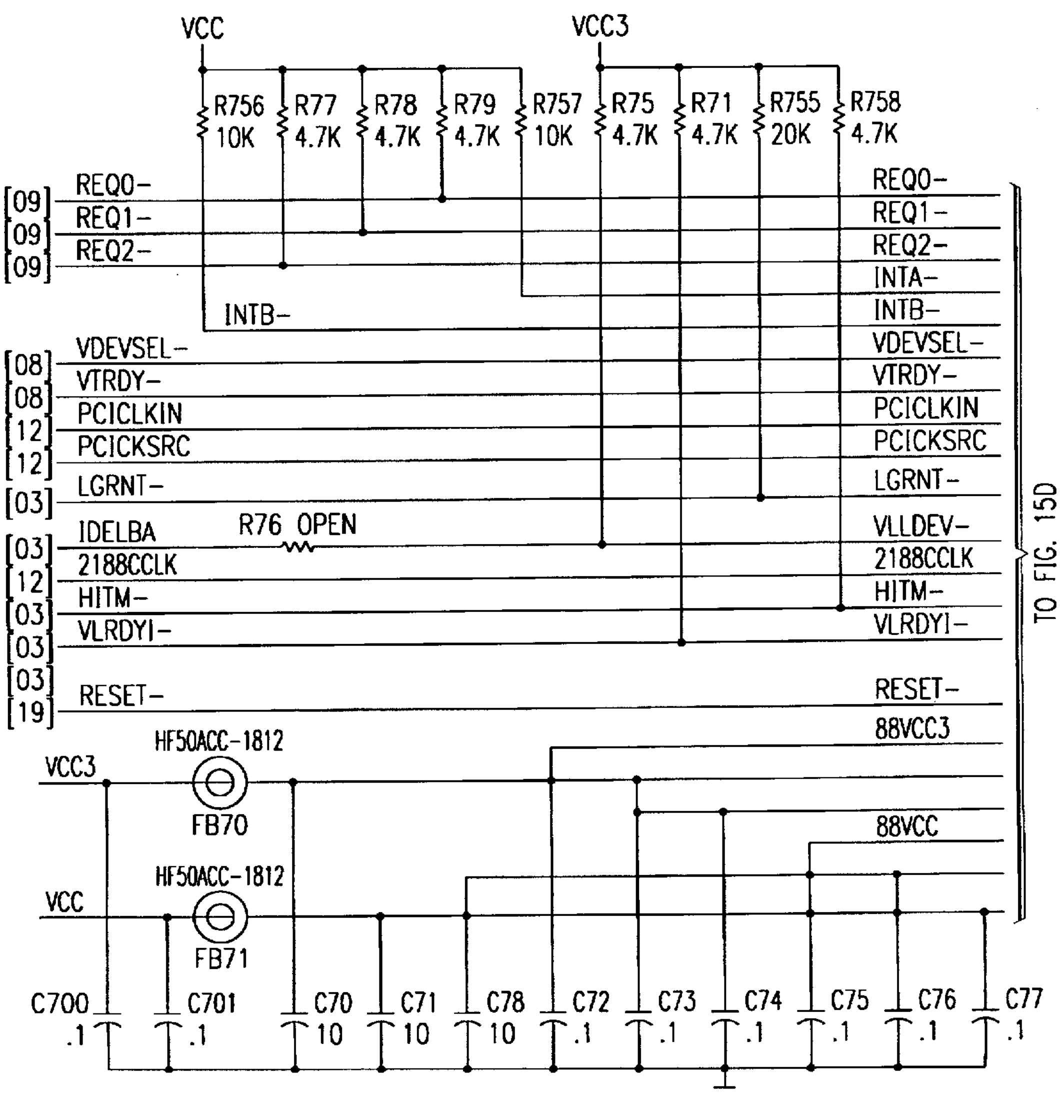

AD<0:31>
[08,10]
CBE<0:3>
[02,03]
FROM FIG. 15A
RP16
20K
RN70
16
VCC
AD13 AD11 AD9 AD7 AD5 AD3 AD1 AD0 AD2 AD4 AD6 AD8 AD10 AD12 AD14
RP16
20K
RN71
16
AD16 AD18 AD20 AD22 AD24 AD26 AD28 AD29 AD27 AD25 AD23 AD21 AD19 AD17 AD15
R743
20K
R744
20K
AD30
AD31
TO FIG. 15E

FIG. 15B

FIG. 15C


120
81
121
80
2188 PINOUT
160
41
1
40


VCC
VCC3
R756 10K
R77 4.7K
R78 4.7K
R79 4.7K
R757 10K
R75 4.7K
R71 4.7K
R755 20K
R758 4.7K
[09] REQ0-
[09] REQ1-
[09] REQ2-
INTB-
[08] VDEVSEL-
[08] VTRDY-
[12] PCICLKIN
[12] PCICKSRC
[03] LGRNT-
[03] IDELBA
R76 OPEN
[12] 2188CCLK
[03] HITM-
[03] VLRDYI-
[03]
[19] RESET-
REQ0-
REQ1-
REQ2-
INTA-
INTB-
VDEVSEL-
VTRDY-
PCICLKIN
PCICKSRC
LGRNT-
VLLDEV-
2188CCLK
HITM-
VLRDYI-
RESET-
88VCC3
88VCC
TO FIG. 15D
HF50ACC-1812
FB70
HF50ACC-1812
FB71
C700 .1
C701 .1
C70 10
C71 10
C78 10
C72 .1
C73 .1
C74 .1
C75 .1
C76 .1
C77 .1

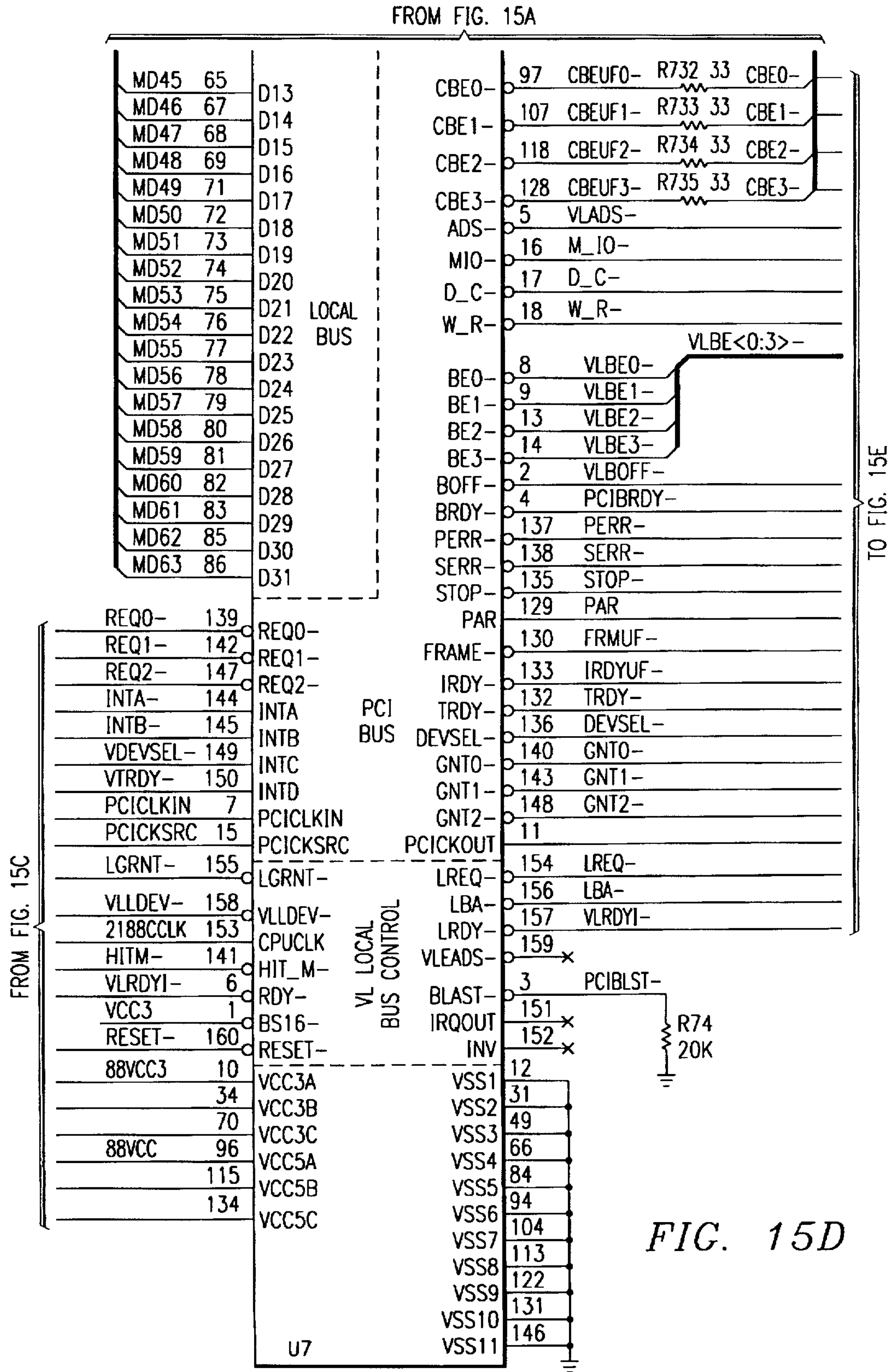
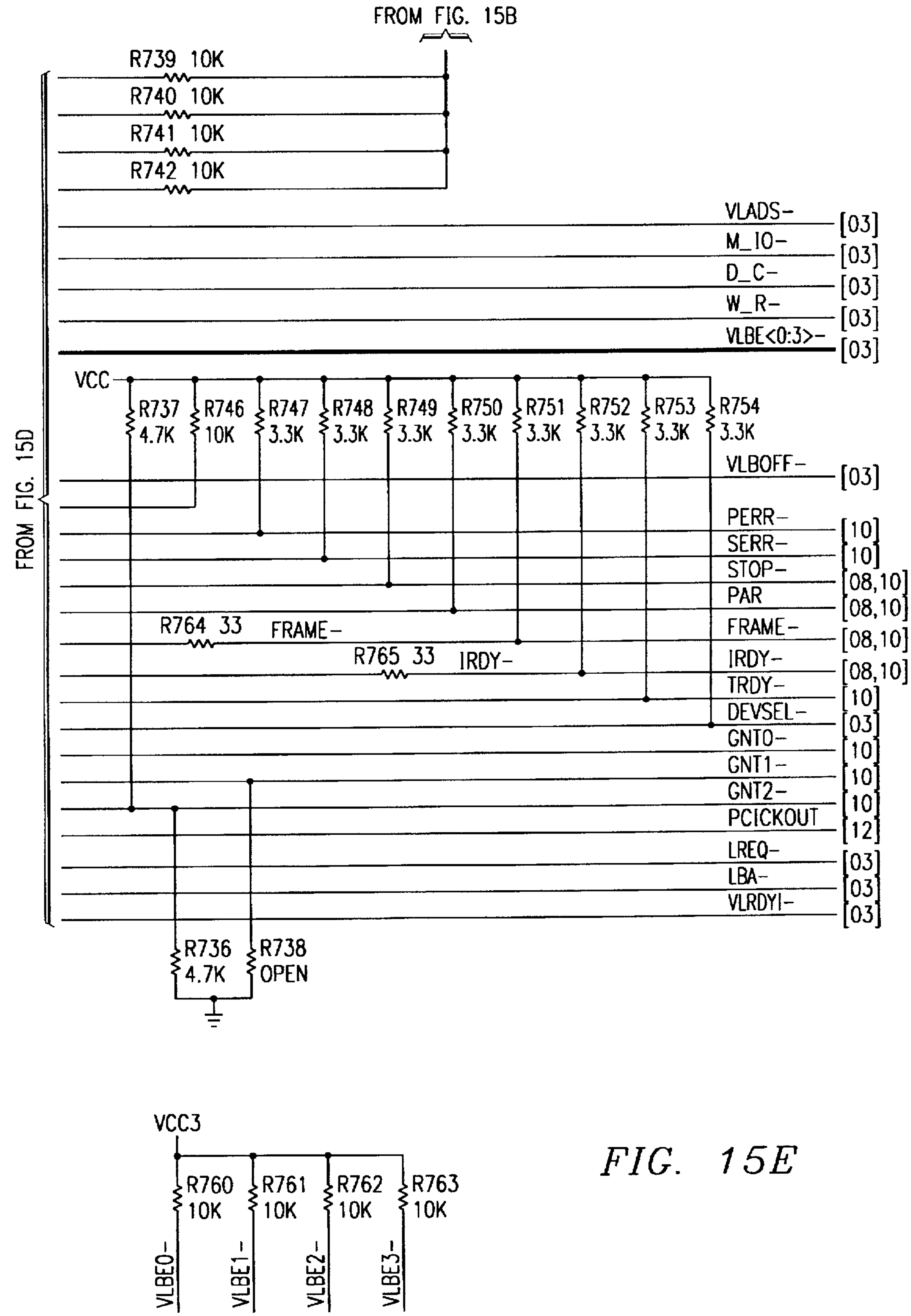
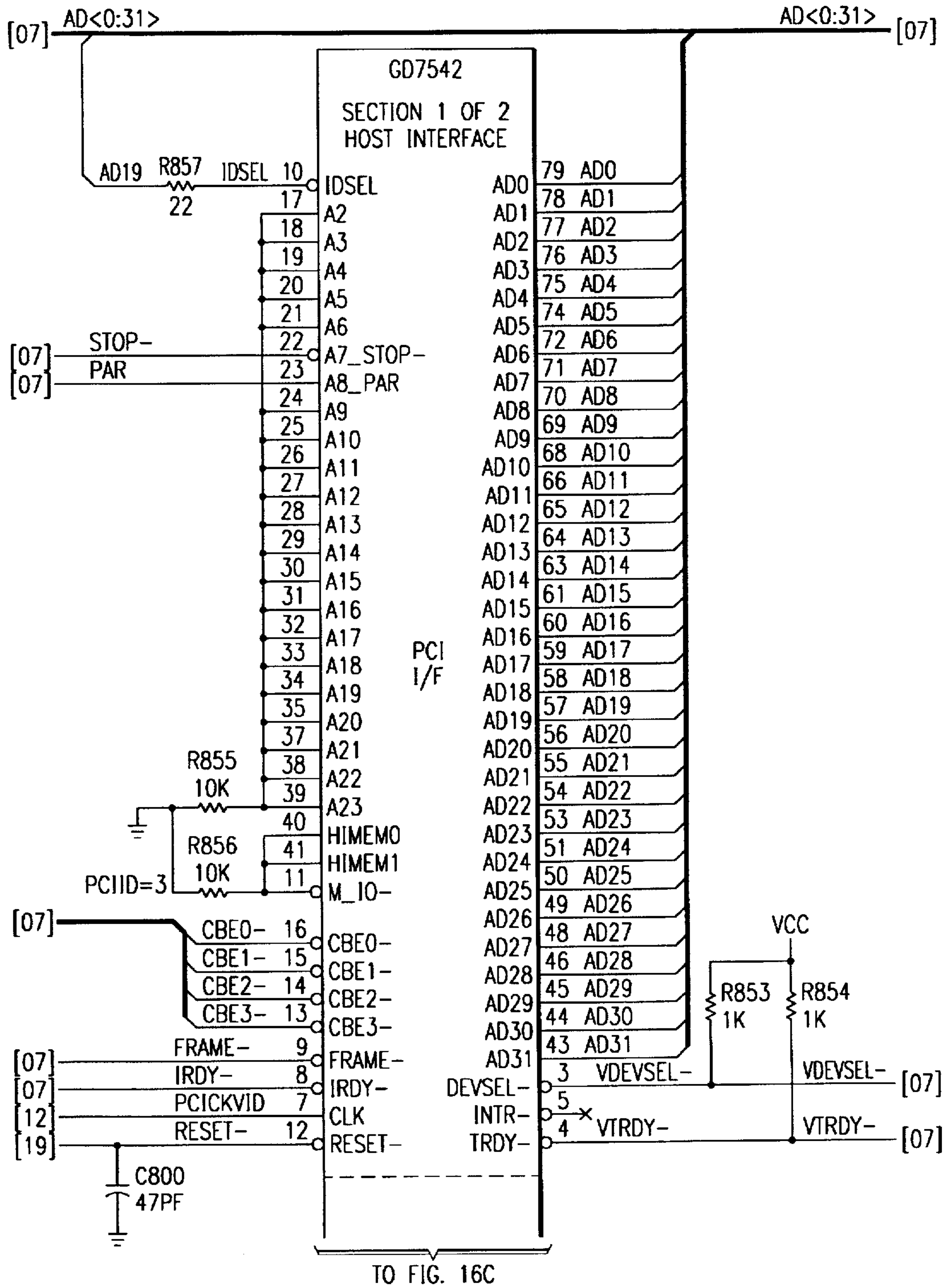

FROM FIG. 15A
TO FIG. 15E
FROM FIG. 15C
LOCAL BUS
PCI BUS
VL LOCAL BUS CONTROL
U7
R732 33
R733 33
R734 33
R735 33
R74 20K
VLBE<0:3>-

FIG. 15D



FROM FIG. 15B
R739 10K
R740 10K
R741 10K
R742 10K
VLADS- [03]
M_IO- [03]
D_C- [03]
W_R- [03]
VLBE<0:3>- [03]
VCC
R737 4.7K
R746 10K
R747 3.3K
R748 3.3K
R749 3.3K
R750 3.3K
R751 3.3K
R752 3.3K
R753 3.3K
R754 3.3K
FROM FIG. 15D
VLBOFF- [03]
PERR- [10]
SERR- [10]
STOP- [08,10]
PAR [08,10]
R764 33
FRAME-
FRAME- [08,10]
R765 33
IRDY-
IRDY- [08,10]
TRDY- [10]
DEVSEL- [03]
GNT0- [10]
GNT1- [10]
GNT2- [10]
PCICKOUT [12]
LREQ- [03]
LBA- [03]
VLRDYI- [03]
R736 4.7K
R738 OPEN
VCC3
R760 10K
R761 10K
R762 10K
R763 10K
VLBE0-
VLBE1-
VLBE2-
VLBE3-

FIG. 15E



[07]
AD<0:31>
AD<0:31>
[07]
GD7542
SECTION 1 OF 2
HOST INTERFACE
AD19
R857
22
IDSEL
10 IDSEL
17 A2
18 A3
19 A4
20 A5
21 A6
22 A7_STOP-
23 A8_PAR
24 A9
25 A10
26 A11
27 A12
28 A13
29 A14
30 A15
31 A16
32 A17
33 A18
34 A19
35 A20
37 A21
38 A22
39 A23
40 HIMEM0
41 HIMEM1
11 M_IO-
STOP-
PAR
[07]
[07]
R855
10K
R856
10K
PCIID=3
[07]
CBE0- 16 CBE0-
CBE1- 15 CBE1-
CBE2- 14 CBE2-
CBE3- 13 CBE3-
FRAME- 9 FRAME-
IRDY- 8 IRDY-
PCICKVID 7 CLK
RESET- 12 RESET-
[07]
[07]
[12]
[19]
C800
47PF
PCI
I/F
AD0 79 AD0
AD1 78 AD1
AD2 77 AD2
AD3 76 AD3
AD4 75 AD4
AD5 74 AD5
AD6 72 AD6
AD7 71 AD7
AD8 70 AD8
AD9 69 AD9
AD10 68 AD10
AD11 66 AD11
AD12 65 AD12
AD13 64 AD13
AD14 63 AD14
AD15 61 AD15
AD16 60 AD16
AD17 59 AD17
AD18 58 AD18
AD19 57 AD19
AD20 56 AD20
AD21 55 AD21
AD22 54 AD22
AD23 53 AD23
AD24 51 AD24
AD25 50 AD25
AD26 49 AD26
AD27 48 AD27
AD28 46 AD28
AD29 45 AD29
AD30 44 AD30
AD31 43 AD31
DEVSEL- 3 VDEVSEL-
INTR- 5
TRDY- 4 VTRDY-
VCC
R853
1K
R854
1K
VDEVSEL-
[07]
VTRDY-
[07]
TO FIG. 16C

FIG. 16A

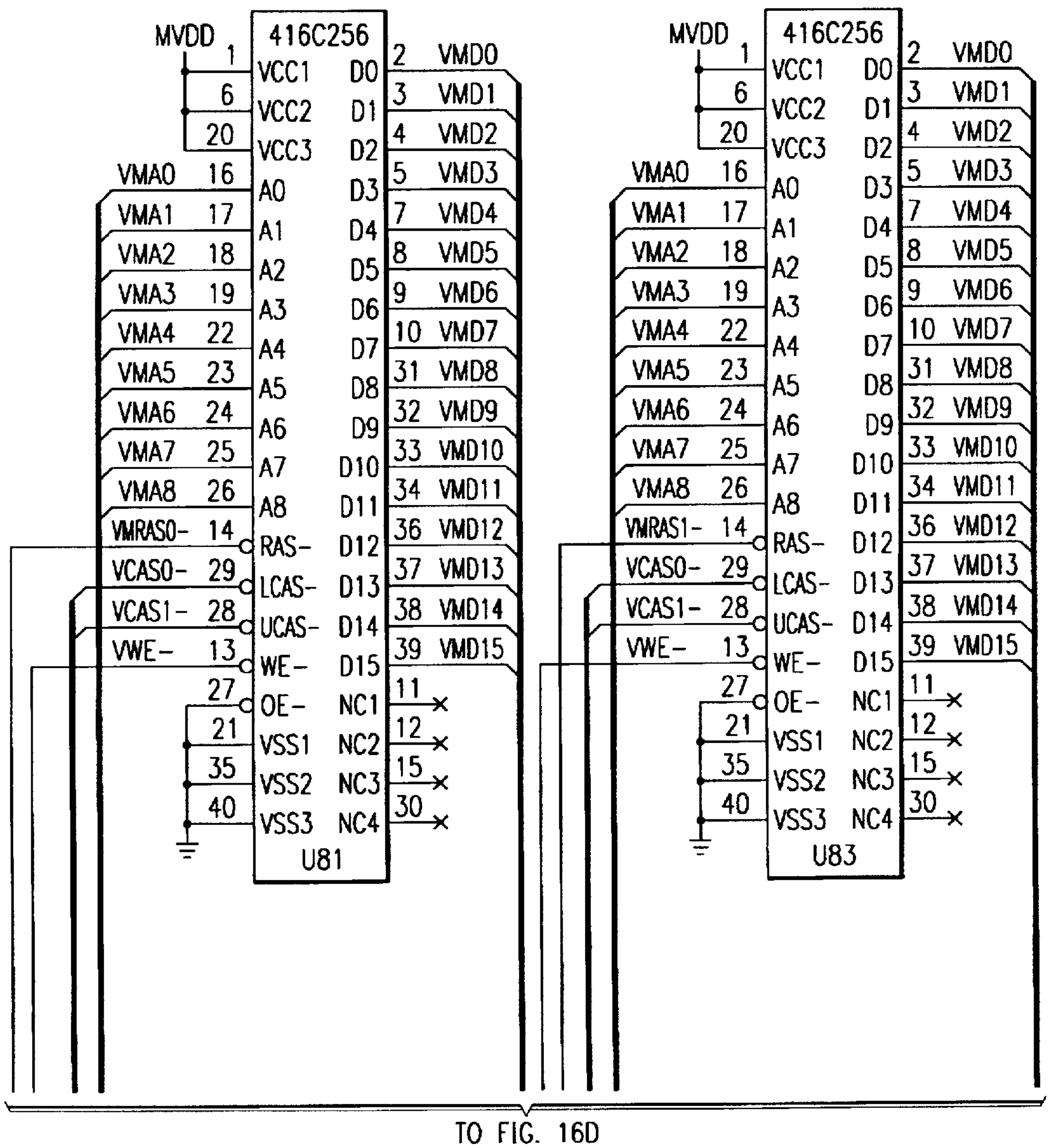
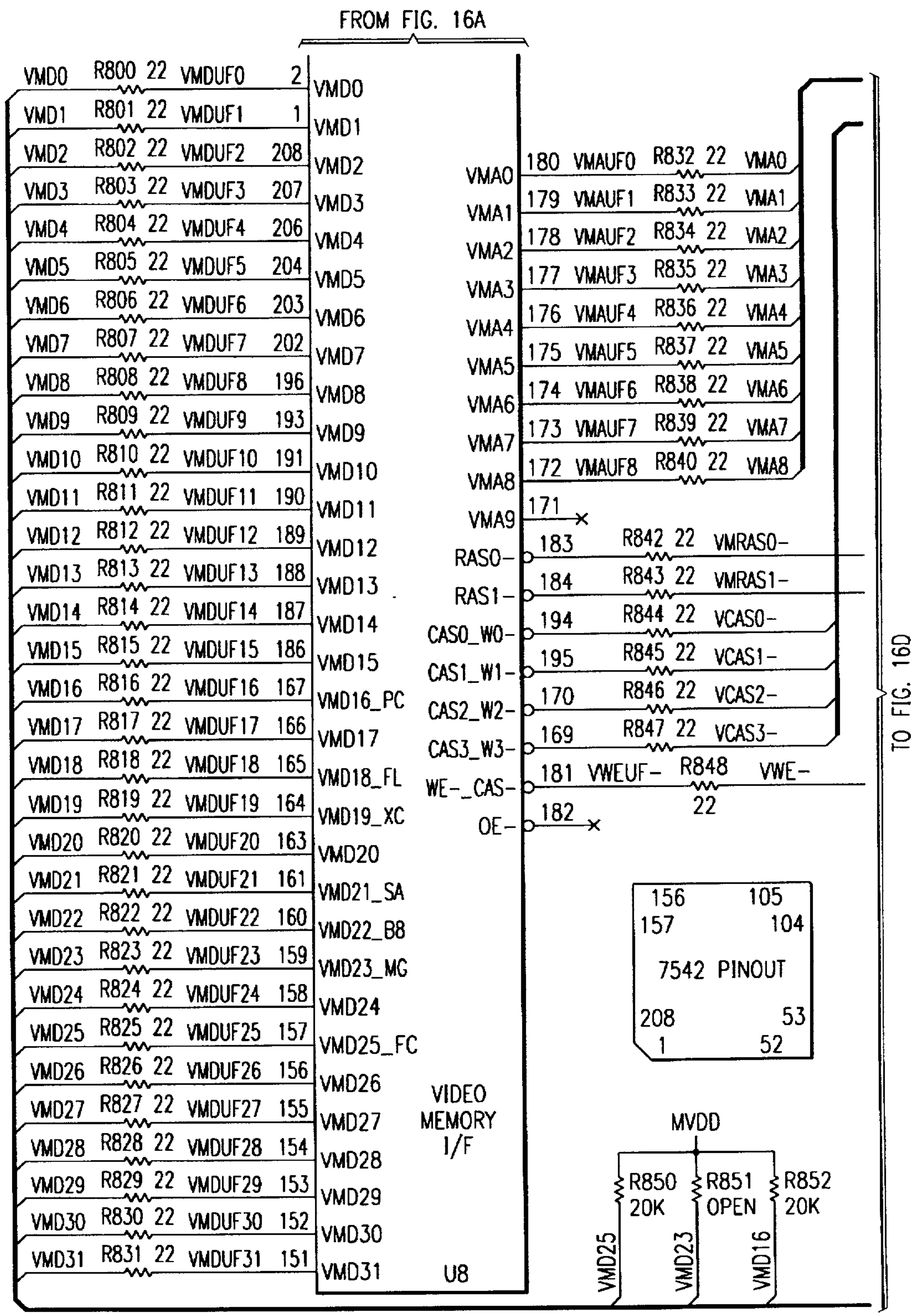

416C256
MVDD
VCC1
VCC2
VCC3
VMA0
VMA1
VMA2
VMA3
VMA4
VMA5
VMA6
VMA7
VMA8
VMRAS0-
VCAS0-
VCAS1-
VWE-
RAS-
LCAS-
UCAS-
WE-
OE-
VSS1
VSS2
VSS3
NC1
NC2
NC3
NC4
U81
VMD0
VMD1
VMD2
VMD3
VMD4
VMD5
VMD6
VMD7
VMD8
VMD9
VMD10
VMD11
VMD12
VMD13
VMD14
VMD15
416C256
VMRAS1-
U83
TO FIG. 16D

FIG. 16B



FROM FIG. 16A
VMD0 R800 22 VMDUF0 2
VMD1 R801 22 VMDUF1 1
VMD2 R802 22 VMDUF2 208
VMD3 R803 22 VMDUF3 207
VMD4 R804 22 VMDUF4 206
VMD5 R805 22 VMDUF5 204
VMD6 R806 22 VMDUF6 203
VMD7 R807 22 VMDUF7 202
VMD8 R808 22 VMDUF8 196
VMD9 R809 22 VMDUF9 193
VMD10 R810 22 VMDUF10 191
VMD11 R811 22 VMDUF11 190
VMD12 R812 22 VMDUF12 189
VMD13 R813 22 VMDUF13 188
VMD14 R814 22 VMDUF14 187
VMD15 R815 22 VMDUF15 186
VMD16 R816 22 VMDUF16 167
VMD17 R817 22 VMDUF17 166
VMD18 R818 22 VMDUF18 165
VMD19 R819 22 VMDUF19 164
VMD20 R820 22 VMDUF20 163
VMD21 R821 22 VMDUF21 161
VMD22 R822 22 VMDUF22 160
VMD23 R823 22 VMDUF23 159
VMD24 R824 22 VMDUF24 158
VMD25 R825 22 VMDUF25 157
VMD26 R826 22 VMDUF26 156
VMD27 R827 22 VMDUF27 155
VMD28 R828 22 VMDUF28 154
VMD29 R829 22 VMDUF29 153
VMD30 R830 22 VMDUF30 152
VMD31 R831 22 VMDUF31 151
VMD0
VMD1
VMD2
VMD3
VMD4
VMD5
VMD6
VMD7
VMD8
VMD9
VMD10
VMD11
VMD12
VMD13
VMD14
VMD15
VMD16_PC
VMD17
VMD18_FL
VMD19_XC
VMD20
VMD21_SA
VMD22_B8
VMD23_MG
VMD24
VMD25_FC
VMD26
VMD27
VMD28
VMD29
VMD30
VMD31
VIDEO MEMORY I/F
U8
VMA0 180 VMAUF0 R832 22 VMA0
VMA1 179 VMAUF1 R833 22 VMA1
VMA2 178 VMAUF2 R834 22 VMA2
VMA3 177 VMAUF3 R835 22 VMA3
VMA4 176 VMAUF4 R836 22 VMA4
VMA5 175 VMAUF5 R837 22 VMA5
VMA6 174 VMAUF6 R838 22 VMA6
VMA7 173 VMAUF7 R839 22 VMA7
VMA8 172 VMAUF8 R840 22 VMA8
VMA9 171
RAS0- 183 R842 22 VMRAS0-
RAS1- 184 R843 22 VMRAS1-
CAS0_W0- 194 R844 22 VCAS0-
CAS1_W1- 195 R845 22 VCAS1-
CAS2_W2- 170 R846 22 VCAS2-
CAS3_W3- 169 R847 22 VCAS3-
WE-_CAS- 181 VWEUF- R848 22 VWE-
OE- 182
156 105
157 104
7542 PINOUT
208 53
1 52
MVDD
R850 20K
R851 OPEN
R852 20K
VMD25
VMD23
VMD16
TO FIG. 16D

FIG. 16C

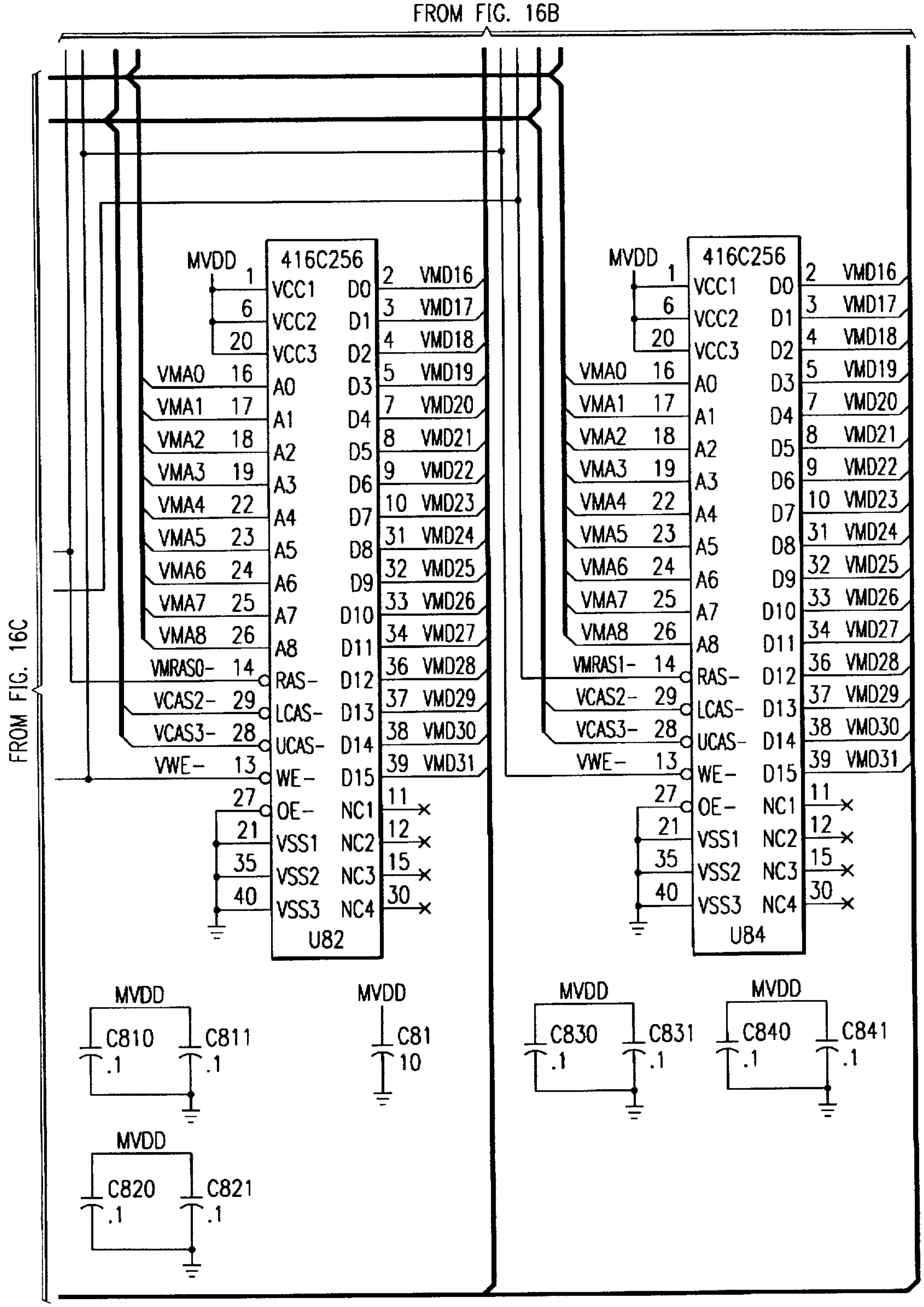
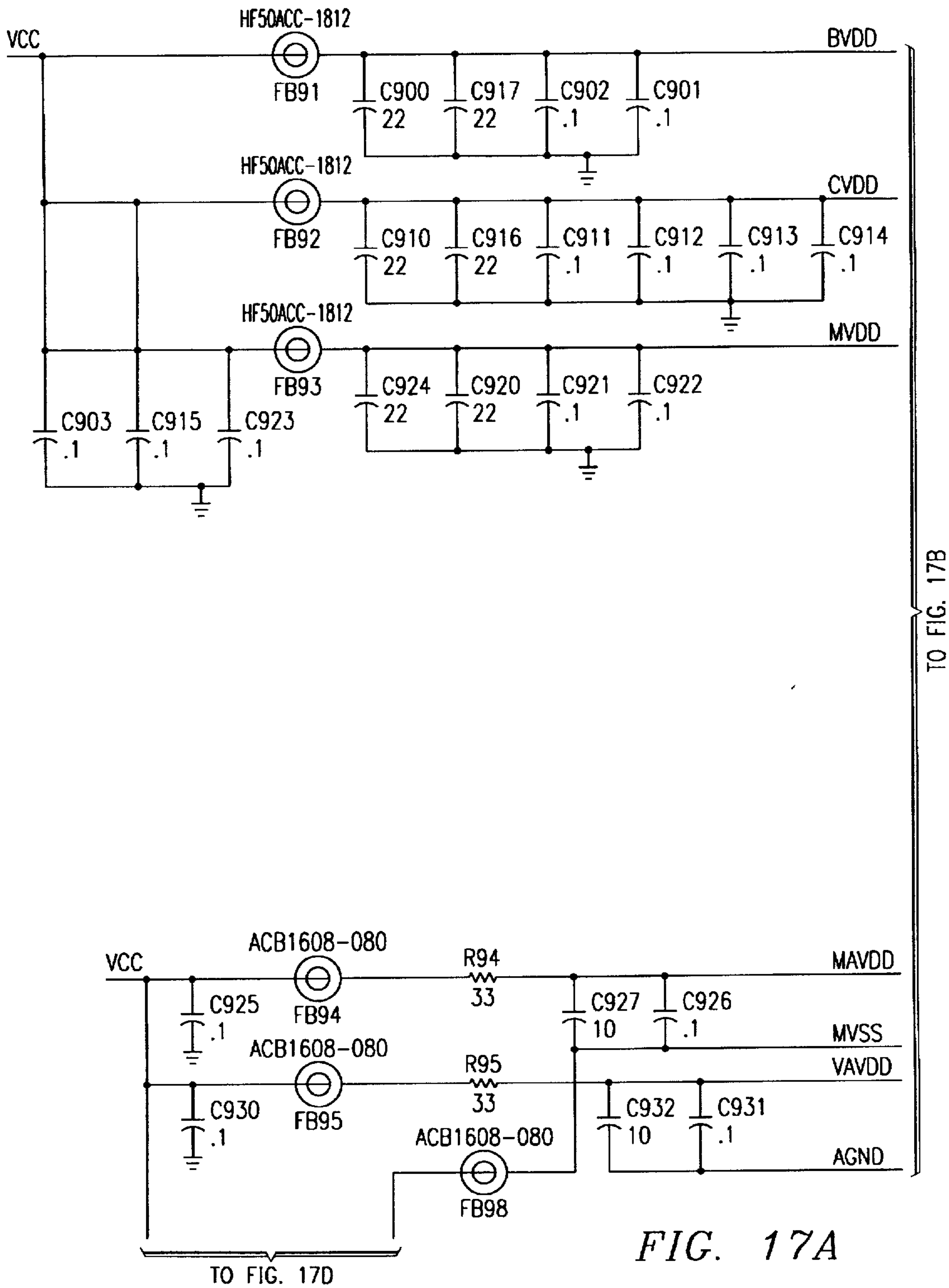
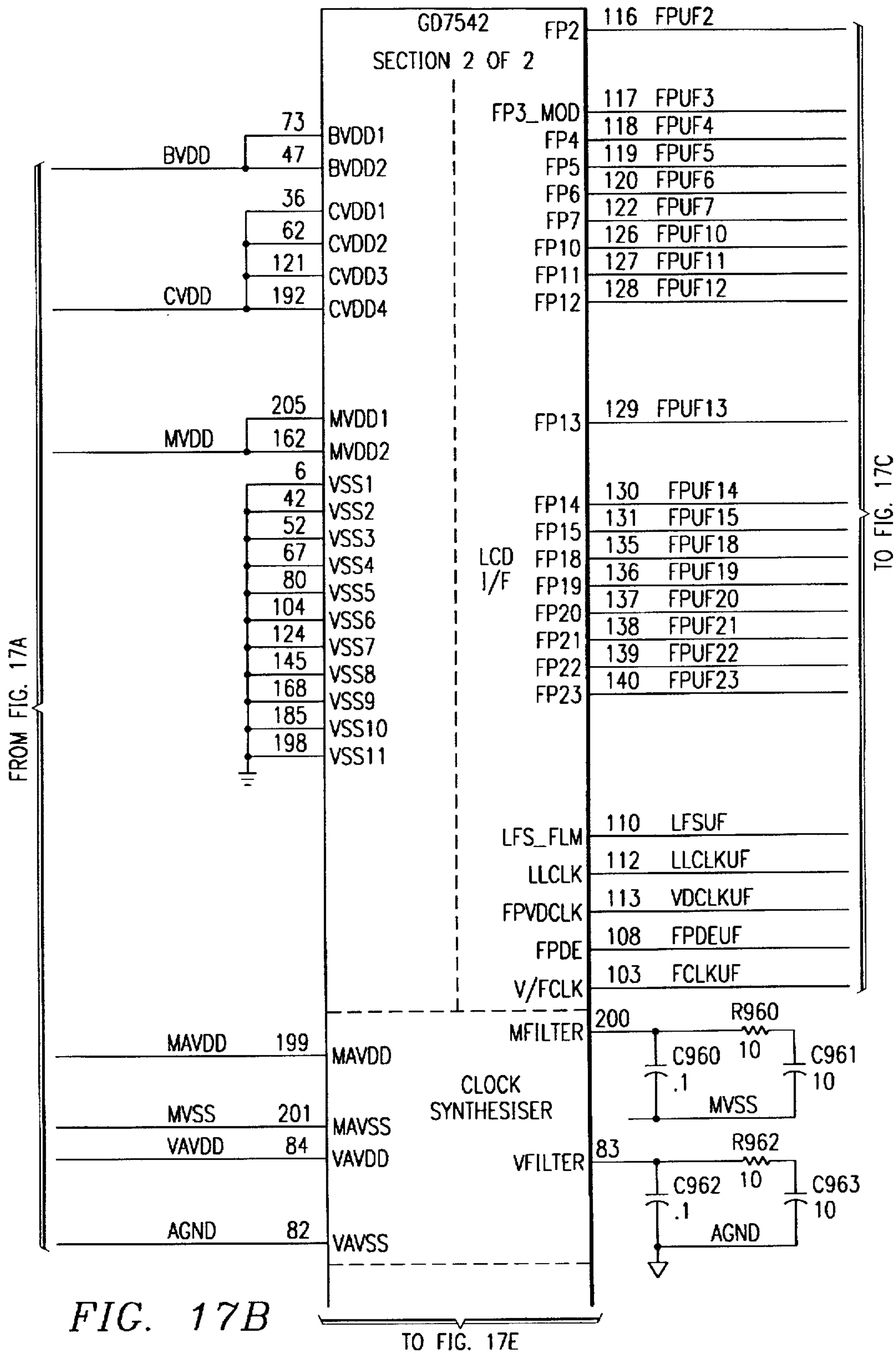

FROM FIG. 16B
FROM FIG. 16C
416C256
MVDD
VCC1 VCC2 VCC3
A0 A1 A2 A3 A4 A5 A6 A7 A8
RAS- LCAS- UCAS- WE- OE-
VSS1 VSS2 VSS3
D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 D10 D11 D12 D13 D14 D15
NC1 NC2 NC3 NC4
VMA0 VMA1 VMA2 VMA3 VMA4 VMA5 VMA6 VMA7 VMA8
VMRAS0- VMRAS1- VCAS2- VCAS3- VWE-
VMD16 VMD17 VMD18 VMD19 VMD20 VMD21 VMD22 VMD23 VMD24 VMD25 VMD26 VMD27 VMD28 VMD29 VMD30 VMD31
U82
U84
C810 .1
C811 .1
C820 .1
C821 .1
C81 10
C830 .1
C831 .1
C840 .1
C841 .1

FIG. 16D



VCC
HF50ACC-1812
FB91
BVDD
C900
22
C917
22
C902
.1
C901
.1
HF50ACC-1812
FB92
CVDD
C910
22
C916
22
C911
.1
C912
.1
C913
.1
C914
.1
HF50ACC-1812
FB93
MVDD
C924
22
C920
22
C921
.1
C922
.1
C903
.1
C915
.1
C923
.1
TO FIG. 17B
VCC
ACB1608-080
FB94
R94
33
MAVDD
C925
.1
C927
10
C926
.1
MVSS
ACB1608-080
FB95
R95
33
VAVDD
C930
.1
C932
10
C931
.1
ACB1608-080
FB98
AGND
TO FIG. 17D

FIG. 17A



GD7542
SECTION 2 OF 2
FROM FIG. 17A
TO FIG. 17C
TO FIG. 17E
BVDD
CVDD
MVDD
MAVDD
MVSS
VAVDD
AGND
LCD I/F
CLOCK SYNTHESISER
FP2 116 FPUF2
FP3_MOD 117 FPUF3
FP4 118 FPUF4
FP5 119 FPUF5
FP6 120 FPUF6
FP7 122 FPUF7
FP10 126 FPUF10
FP11 127 FPUF11
FP12 128 FPUF12
FP13 129 FPUF13
FP14 130 FPUF14
FP15 131 FPUF15
FP18 135 FPUF18
FP19 136 FPUF19
FP20 137 FPUF20
FP21 138 FPUF21
FP22 139 FPUF22
FP23 140 FPUF23
LFS_FLM 110 LFSUF
LLCLK 112 LLCLKUF
FPVDCLK 113 VDCLKUF
FPDE 108 FPDEUF
V/FCLK 103 FCLKUF
73 BVDD1
47 BVDD2
36 CVDD1
62 CVDD2
121 CVDD3
192 CVDD4
205 MVDD1
162 MVDD2
6 VSS1
42 VSS2
52 VSS3
67 VSS4
80 VSS5
104 VSS6
124 VSS7
145 VSS8
168 VSS9
185 VSS10
198 VSS11
199 MAVDD
201 MAVSS
84 VAVDD
82 VAVSS
MFILTER 200
R960 10
C960 .1
C961 10
MVSS
VFILTER 83
R962 10
C962 .1
C963 10
AGND

FIG. 17B

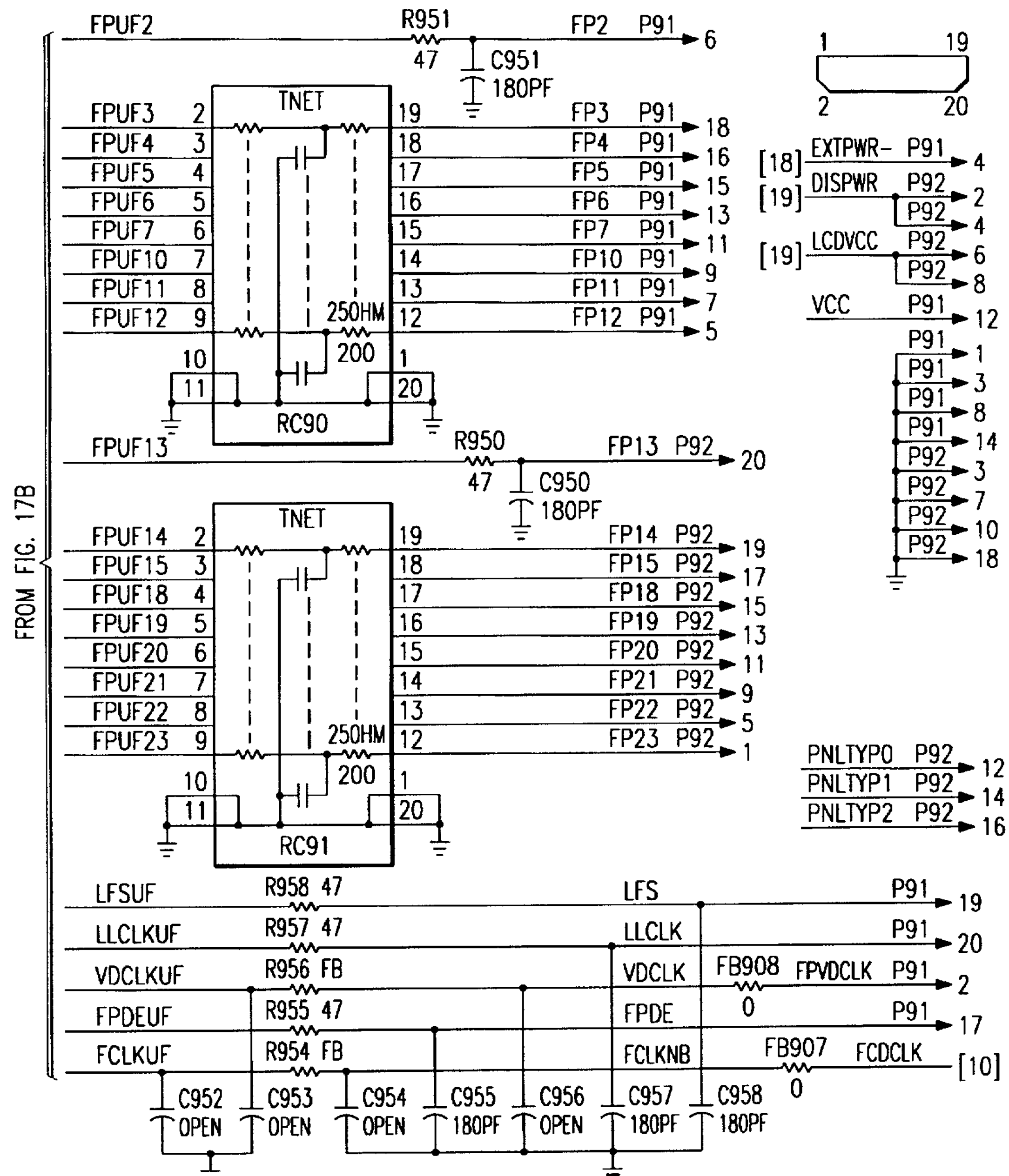
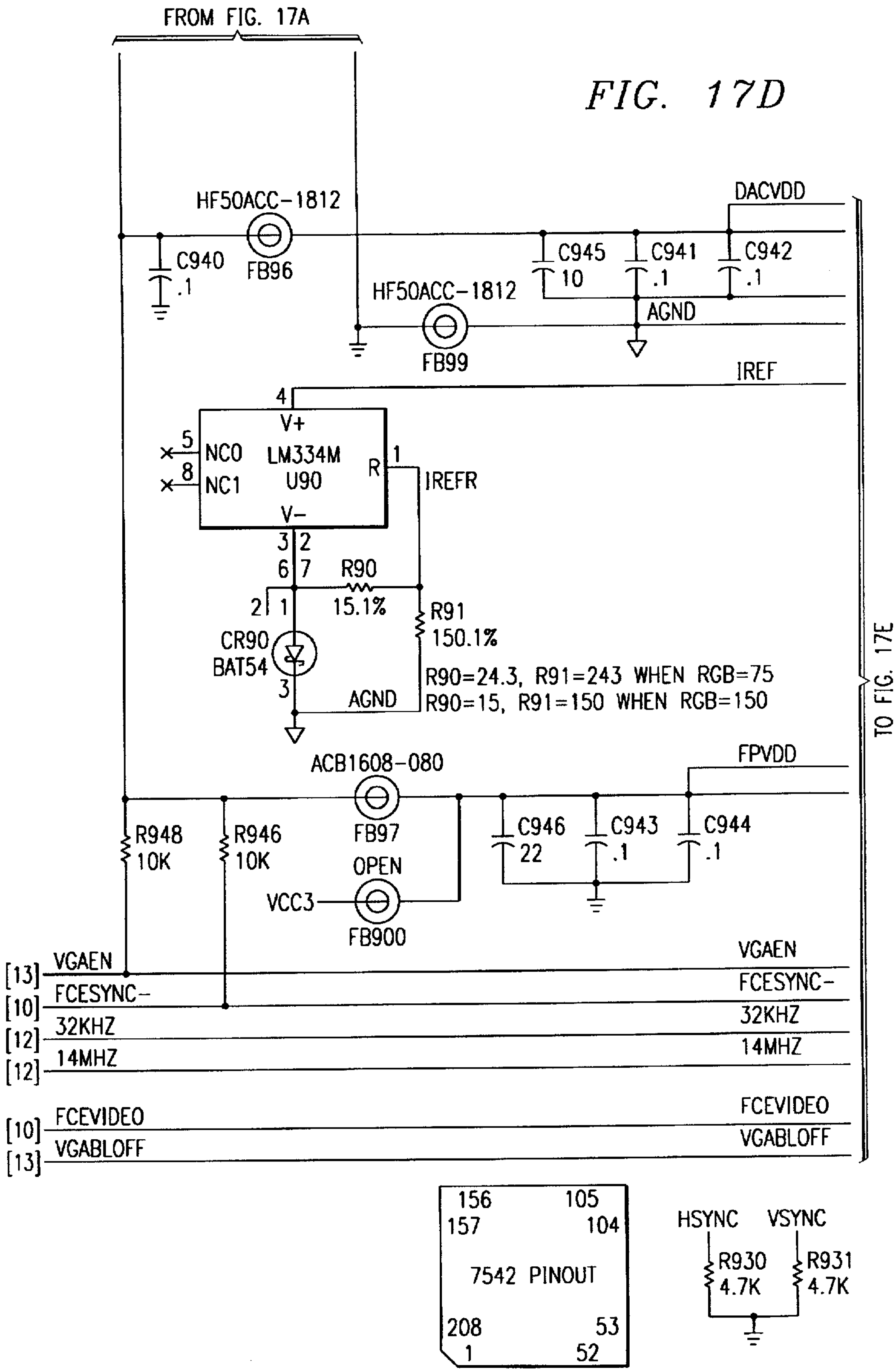
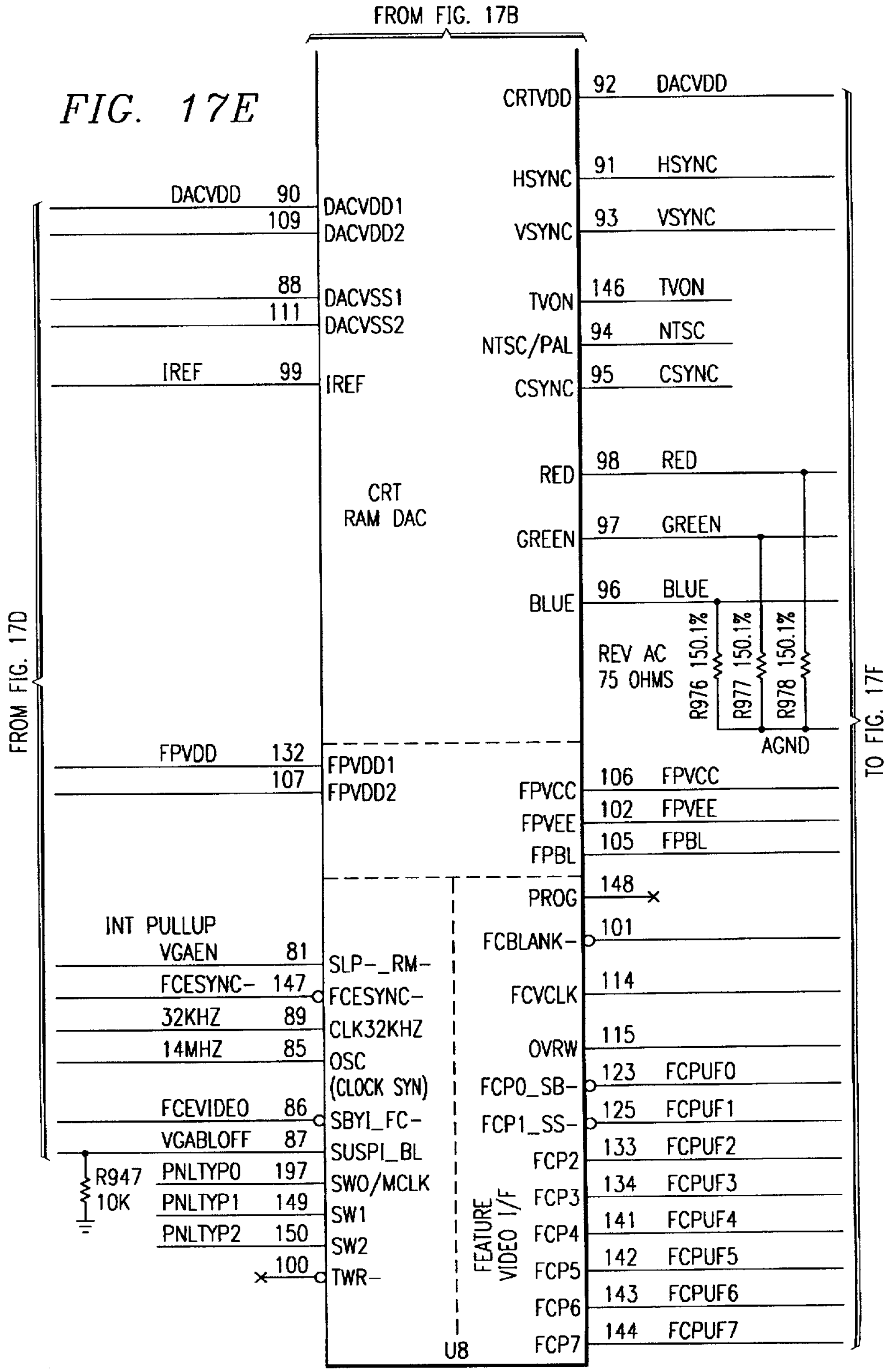

FROM FIG. 17B
FPUF2
R951 47
C951 180PF
FP2 P91 6
TNET
RC90
FPUF3 FPUF4 FPUF5 FPUF6 FPUF7 FPUF10 FPUF11 FPUF12
FP3 P91 18
FP4 P91 16
FP5 P91 15
FP6 P91 13
FP7 P91 11
FP10 P91 9
FP11 P91 7
FP12 P91 5
25OHM
200
FPUF13
R950 47
C950 180PF
FP13 P92 20
TNET
RC91
FPUF14 FPUF15 FPUF18 FPUF19 FPUF20 FPUF21 FPUF22 FPUF23
FP14 P92 19
FP15 P92 17
FP18 P92 15
FP19 P92 13
FP20 P92 11
FP21 P92 9
FP22 P92 5
FP23 P92 1
[18] EXTPWR- P91 4
[19] DISPWR P92 2
P92 4
[19] LCDVCC P92 6
P92 8
VCC P91 12
PNLTYP0 P92 12
PNLTYP1 P92 14
PNLTYP2 P92 16
LFSUF R958 47 LFS P91 19
LLCLKUF R957 47 LLCLK P91 20
VDCLKUF R956 FB VDCLK FB908 0 FPVDCLK P91 2
FPDEUF R955 47 FPDE P91 17
FCLKUF R954 FB FCLKNB FB907 0 FCDCLK [10]
C952 OPEN
C953 OPEN
C954 OPEN
C955 180PF
C956 OPEN
C957 180PF
C958 180PF

FIG. 17C



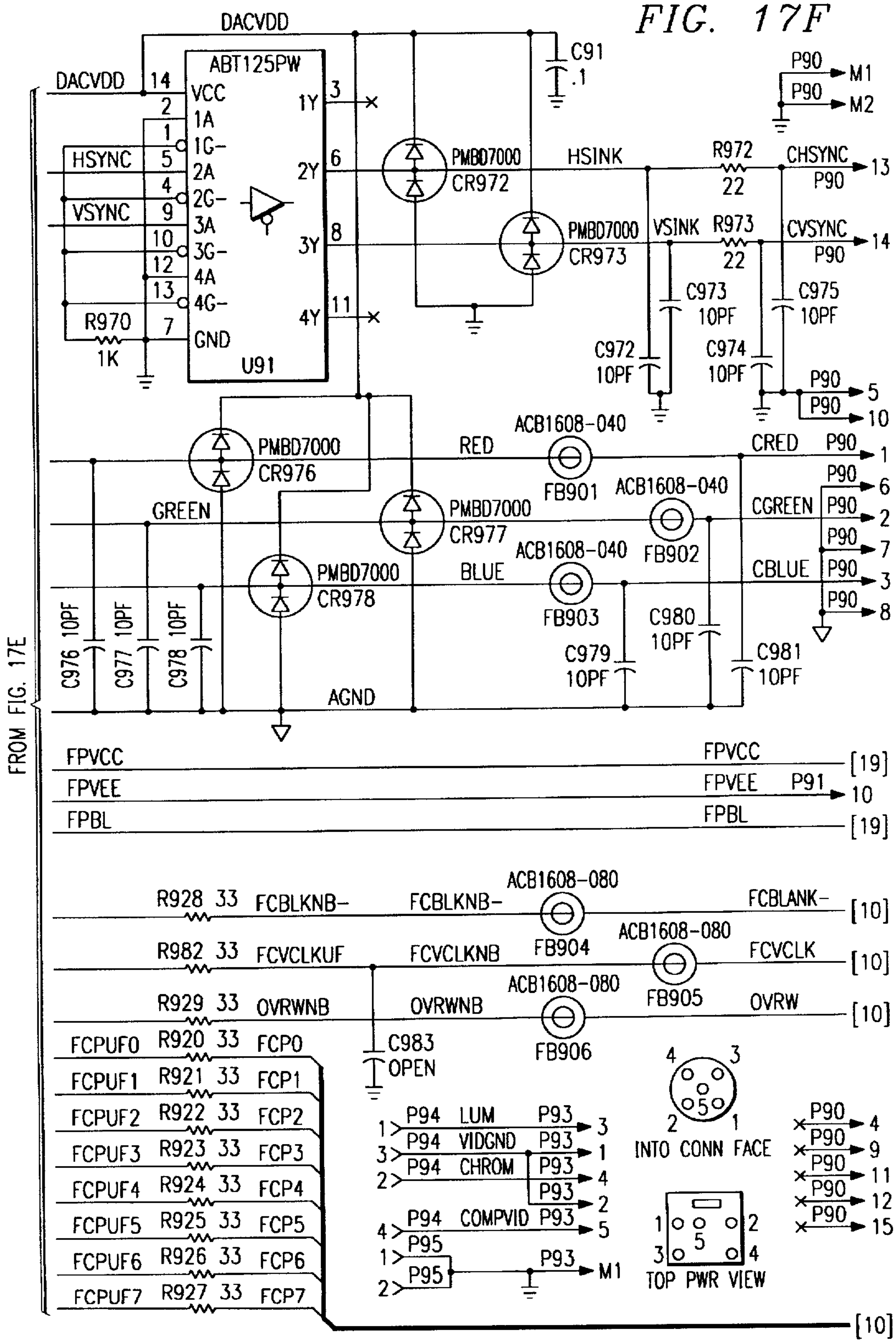
FROM FIG. 17A
FIG. 17D
HF50ACC-1812
FB96
C940 .1
HF50ACC-1812
FB99
DACVDD
C945 10
C941 .1
C942 .1
AGND
IREF
V+
NC0
NC1
LM334M
U90
R
V-
IREFR
R90 15.1%
R91 150.1%
CR90 BAT54
AGND
R90=24.3, R91=243 WHEN RGB=75
R90=15, R91=150 WHEN RGB=150
TO FIG. 17E
ACB1608-080
FB97
OPEN
VCC3
FB900
FPVDD
C946 22
C943 .1
C944 .1
R948 10K
R946 10K
[13] VGAEN
[10] FCESYNC-
[12] 32KHZ
[12] 14MHZ
[10] FCEVIDEO
[13] VGABLOFF
VGAEN
FCESYNC-
32KHZ
14MHZ
FCEVIDEO
VGABLOFF
156 105
157 104
7542 PINOUT
208 53
1 52
HSYNC VSYNC
R930 4.7K
R931 4.7K

FIG. 17E



FROM FIG. 17B
FROM FIG. 17D
TO FIG. 17F
CRT
RAM DAC
DACVDD 90 DACVDD1
109 DACVDD2
88 DACVSS1
111 DACVSS2
IREF 99 IREF
FPVDD 132 FPVDD1
107 FPVDD2
INT PULLUP
VGAEN 81 SLP-_RM-
FCESYNC- 147 FCESYNC-
32KHZ 89 CLK32KHZ
14MHZ 85 OSC
(CLOCK SYN)
FCEVIDEO 86 SBYI_FC-
VGABLOFF 87 SUSPI_BL
R947
10K
PNLTYP0 197 SWO/MCLK
PNLTYP1 149 SW1
PNLTYP2 150 SW2
100 TWR-
CRTVDD 92 DACVDD
HSYNC 91 HSYNC
VSYNC 93 VSYNC
TVON 146 TVON
NTSC/PAL 94 NTSC
CSYNC 95 CSYNC
RED 98 RED
GREEN 97 GREEN
BLUE 96 BLUE
REV AC
75 OHMS
R976 150.1%
R977 150.1%
R978 150.1%
AGND
FPVCC 106 FPVCC
FPVEE 102 FPVEE
FPBL 105 FPBL
PROG 148
FCBLANK- 101
FCVCLK 114
OVRW 115
FCP0_SB- 123 FCPUF0
FCP1_SS- 125 FCPUF1
FCP2 133 FCPUF2
FCP3 134 FCPUF3
FCP4 141 FCPUF4
FCP5 142 FCPUF5
FCP6 143 FCPUF6
FCP7 144 FCPUF7
FEATURE
VIDEO I/F
U8

FIG. 17F


DACVDD
ABT125PW
VCC
1A
1G-
2A
2G-
3A
3G-
4A
4G-
GND
1Y
2Y
3Y
4Y
U91
HSYNC
VSYNC
R970
1K
C91
.1
P90
M1
M2
PMBD7000
CR972
CR973
HSINK
VSINK
R972
22
R973
CHSYNC
CVSYNC
C973
10PF
C975
C972
C974
CR976
CR977
CR978
RED
GREEN
BLUE
ACB1608-040
FB901
FB902
FB903
CRED
CGREEN
CBLUE
C976 10PF
C977 10PF
C978 10PF
C979
C980
C981
AGND
FROM FIG. 17E
FPVCC
FPVEE
FPBL
P91
[19]
R928 33
FCBLKNB-
ACB1608-080
FB904
FCBLANK-
[10]
R982 33
FCVCLKUF
FCVCLKNB
FB905
FCVCLK
R929 33
OVRWNB
FB906
OVRW
C983
OPEN
FCPUF0 R920 33 FCP0
FCPUF1 R921 33 FCP1
FCPUF2 R922 33 FCP2
FCPUF3 R923 33 FCP3
FCPUF4 R924 33 FCP4
FCPUF5 R925 33 FCP5
FCPUF6 R926 33 FCP6
FCPUF7 R927 33 FCP7
P94 LUM P93
P94 VIDGND P93
P94 CHROM P93
P94 COMPVID P93
P95
INTO CONN FACE
TOP PWR VIEW

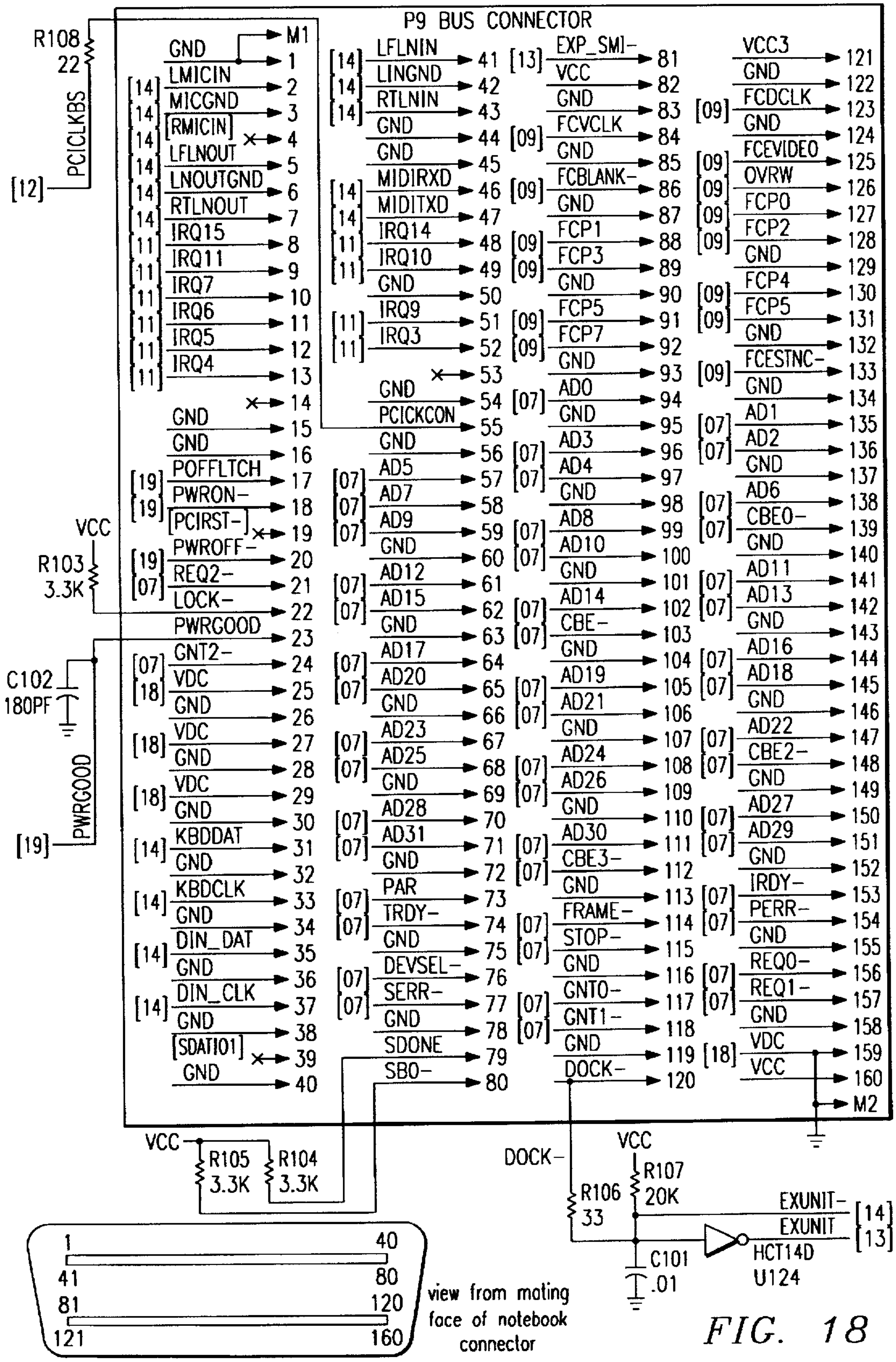
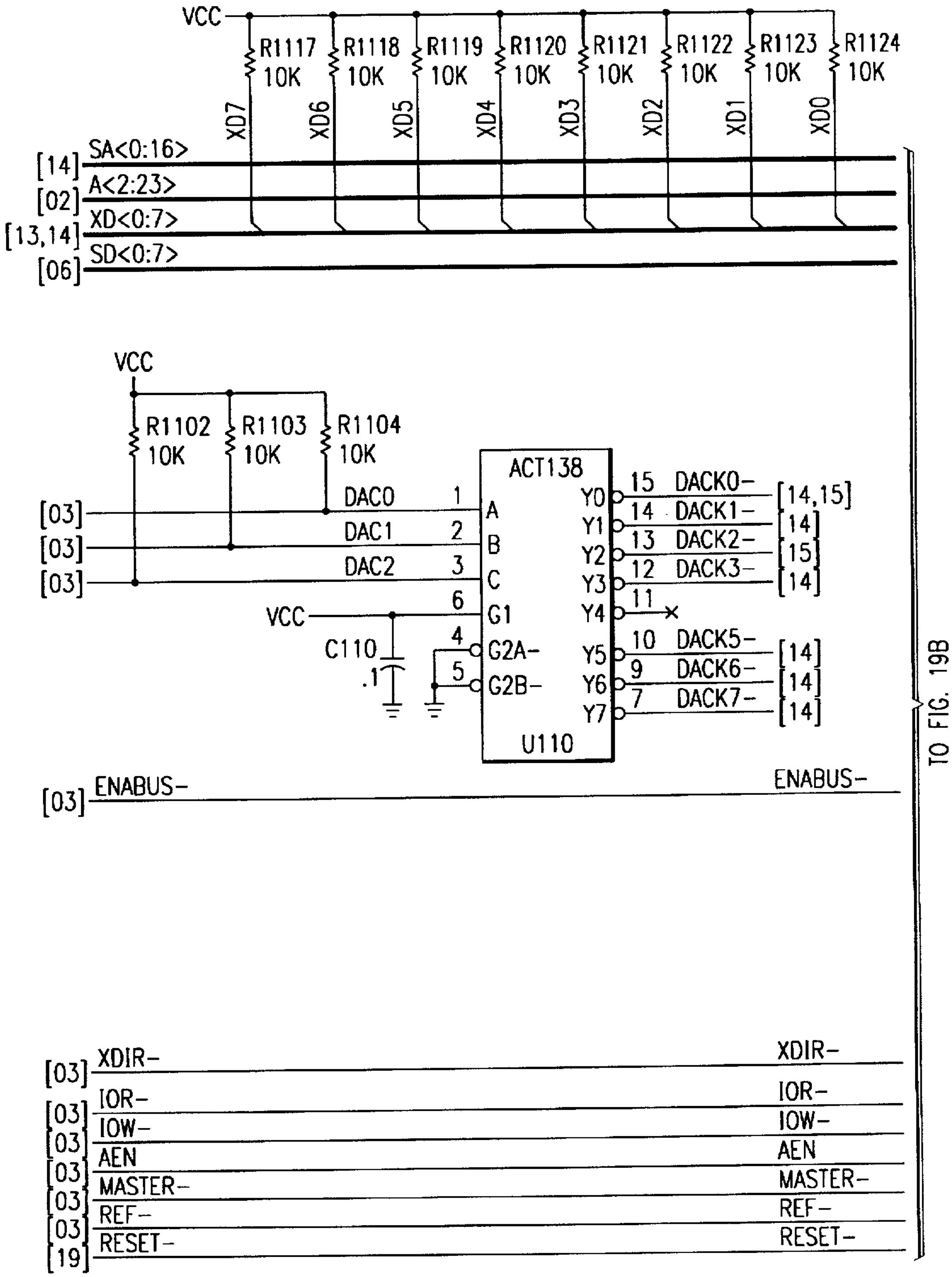

P9 BUS CONNECTOR
R108
22
PCICLKBS
VCC
R103
3.3K
C102
180PF
PWRGOOD
GND
LMICIN
MICGND
RMICIN
LFLNOUT
LNOUTGND
RTLNOUT
IRQ15
IRQ11
IRQ7
IRQ6
IRQ5
IRQ4
POFFLTCH
PWRON-
PCIRST-
PWROFF-
REQ2-
LOCK-
GNT2-
VDC
KBDDAT
KBDCLK
DIN_DAT
DIN_CLK
SDATI01
LFLNIN
LINGND
RTLNIN
MIDIRXD
MIDITXD
IRQ14
IRQ10
IRQ9
IRQ3
PCICKCON
AD5
AD7
AD9
AD12
AD15
AD17
AD20
AD23
AD25
AD28
AD31
PAR
TRDY-
DEVSEL-
SERR-
SDONE
SBO-
EXP_SMI-
VCC
FCVCLK
FCBLANK-
FCP1
FCP3
FCP5
FCP7
AD0
AD3
AD4
AD8
AD10
AD14
CBE-
AD19
AD21
AD24
AD26
AD30
CBE3-
FRAME-
STOP-
GNT0-
GNT1-
DOCK-
VCC3
FCDCLK
FCEVIDEO
OVRW
FCP0
FCP2
FCP4
FCESTNC-
AD1
AD2
AD6
CBE0-
AD11
AD13
AD16
AD18
AD22
CBE2-
AD27
AD29
IRDY-
PERR-
REQ0-
REQ1-
M1
M2
R105
3.3K
R104
3.3K
R106
33
R107
20K
C101
.01
EXUNIT-
EXUNIT
HCT14D
U124
view from mating face of notebook connector

FIG. 18



VCC
R1117 10K
R1118 10K
R1119 10K
R1120 10K
R1121 10K
R1122 10K
R1123 10K
R1124 10K
XD7
XD6
XD5
XD4
XD3
XD2
XD1
XD0
[14] SA<0:16>
[02] A<2:23>
[13,14] XD<0:7>
[06] SD<0:7>
VCC
R1102 10K
R1103 10K
R1104 10K
ACT138
[03] DAC0 1 A
[03] DAC1 2 B
[03] DAC2 3 C
VCC 6 G1
C110 .1
4 G2A-
5 G2B-
Y0 15 DACK0- [14,15]
Y1 14 DACK1- [14]
Y2 13 DACK2- [15]
Y3 12 DACK3- [14]
Y4 11
Y5 10 DACK5- [14]
Y6 9 DACK6- [14]
Y7 7 DACK7- [14]
U110
[03] ENABUS-
ENABUS-
TO FIG. 19B
[03] XDIR-
XDIR-
[03] IOR-
IOR-
[03] IOW-
IOW-
[03] AEN
AEN
[03] MASTER-
MASTER-
[03] REF-
REF-
[19] RESET-
RESET-

FIG. 19A

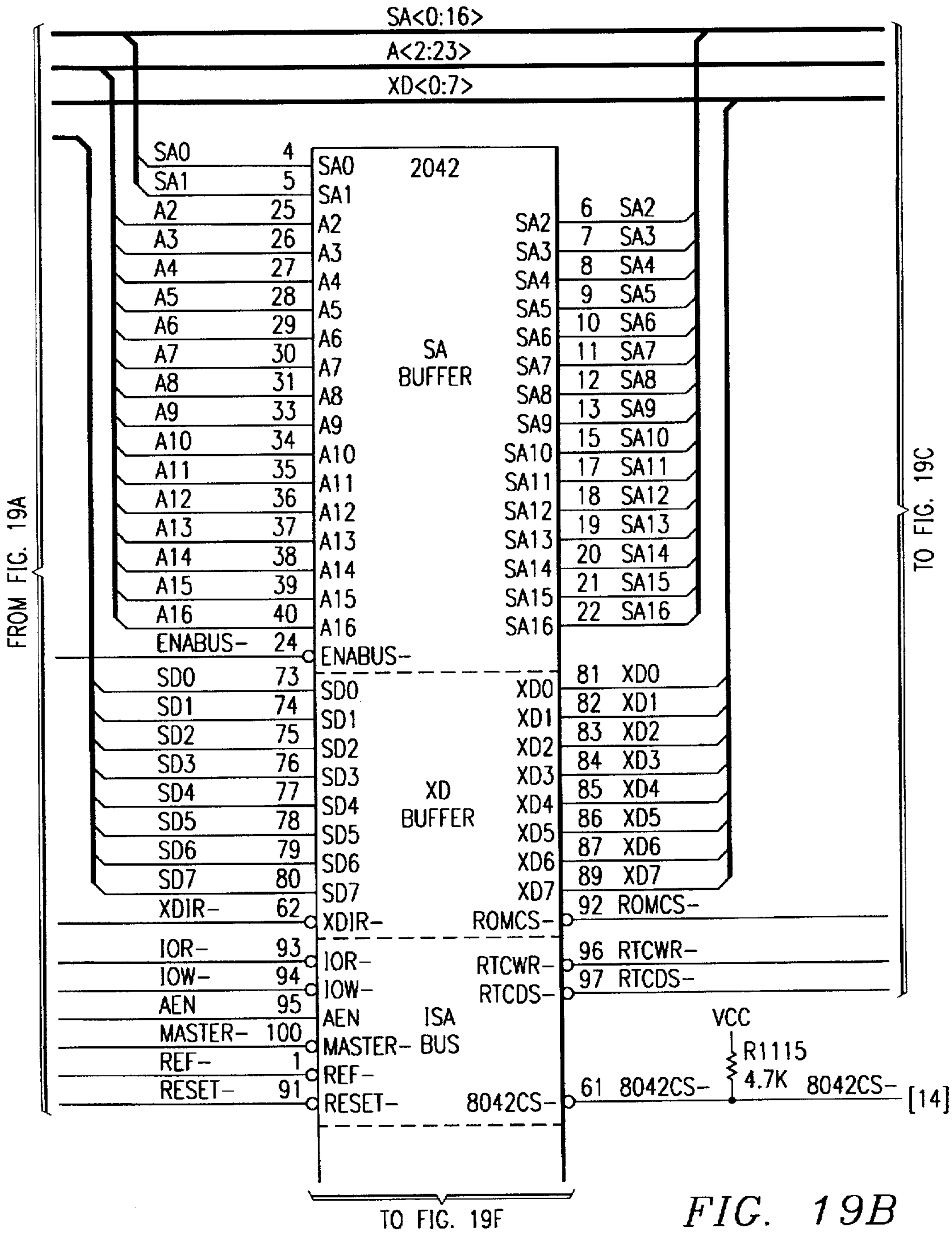

SA<0:16>
A<2:23>
XD<0:7>
FROM FIG. 19A
TO FIG. 19C
2042
SA BUFFER
XD BUFFER
ISA BUS
SA0 4 SA0
SA1 5 SA1
A2 25 A2
A3 26 A3
A4 27 A4
A5 28 A5
A6 29 A6
A7 30 A7
A8 31 A8
A9 33 A9
A10 34 A10
A11 35 A11
A12 36 A12
A13 37 A13
A14 38 A14
A15 39 A15
A16 40 A16
ENABUS- 24 ENABUS-
SA2 6 SA2
SA3 7 SA3
SA4 8 SA4
SA5 9 SA5
SA6 10 SA6
SA7 11 SA7
SA8 12 SA8
SA9 13 SA9
SA10 15 SA10
SA11 17 SA11
SA12 18 SA12
SA13 19 SA13
SA14 20 SA14
SA15 21 SA15
SA16 22 SA16
SD0 73 SD0
SD1 74 SD1
SD2 75 SD2
SD3 76 SD3
SD4 77 SD4
SD5 78 SD5
SD6 79 SD6
SD7 80 SD7
XDIR- 62 XDIR-
XD0 81 XD0
XD1 82 XD1
XD2 83 XD2
XD3 84 XD3
XD4 85 XD4
XD5 86 XD5
XD6 87 XD6
XD7 89 XD7
ROMCS- 92 ROMCS-
IOR- 93 IOR-
IOW- 94 IOW-
AEN 95 AEN
MASTER- 100 MASTER-
REF- 1 REF-
RESET- 91 RESET-
RTCWR- 96 RTCWR-
RTCDS- 97 RTCDS-
VCC
R1115
4.7K
8042CS- 61 8042CS-
8042CS- [14]
TO FIG. 19F

FIG. 19B

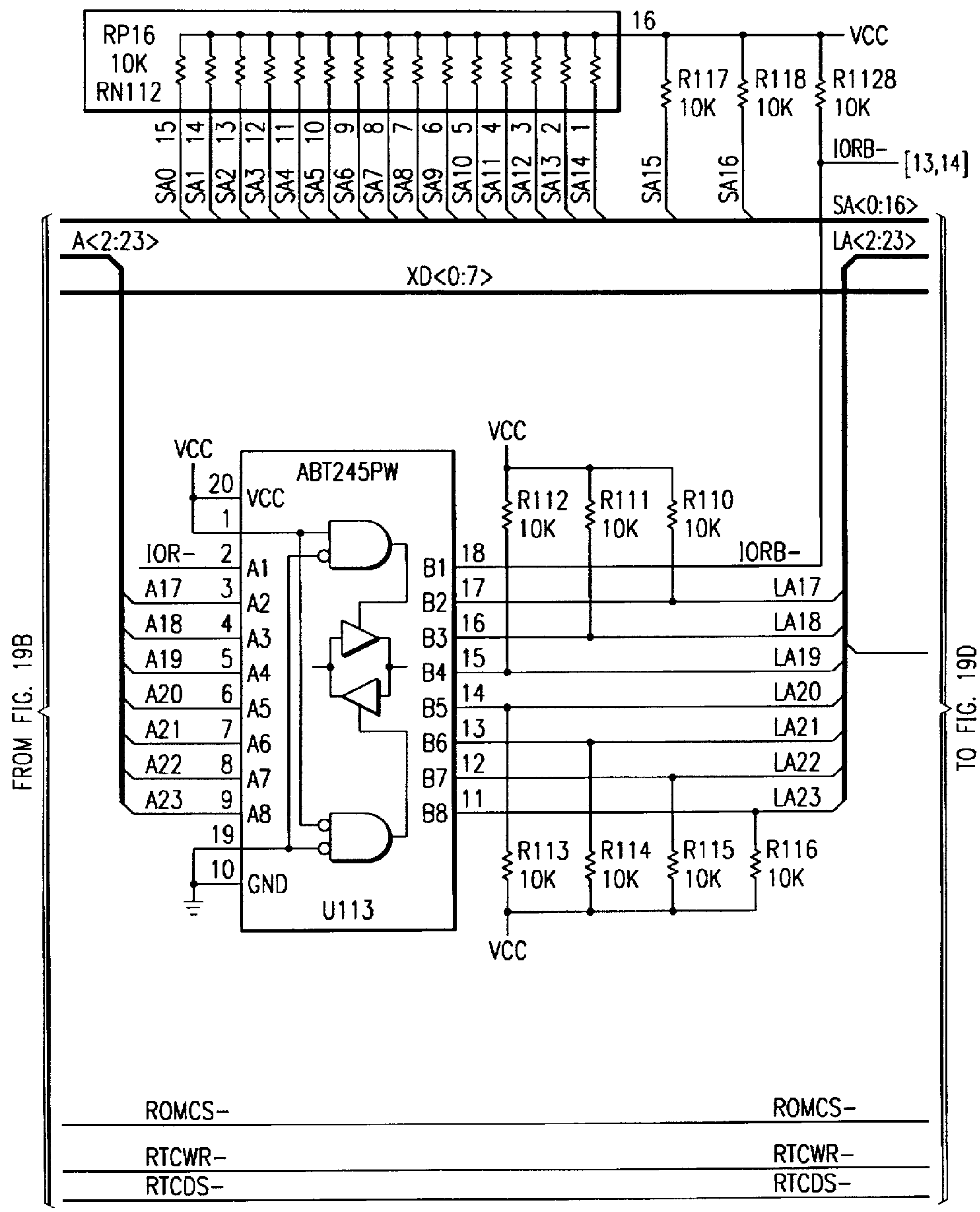
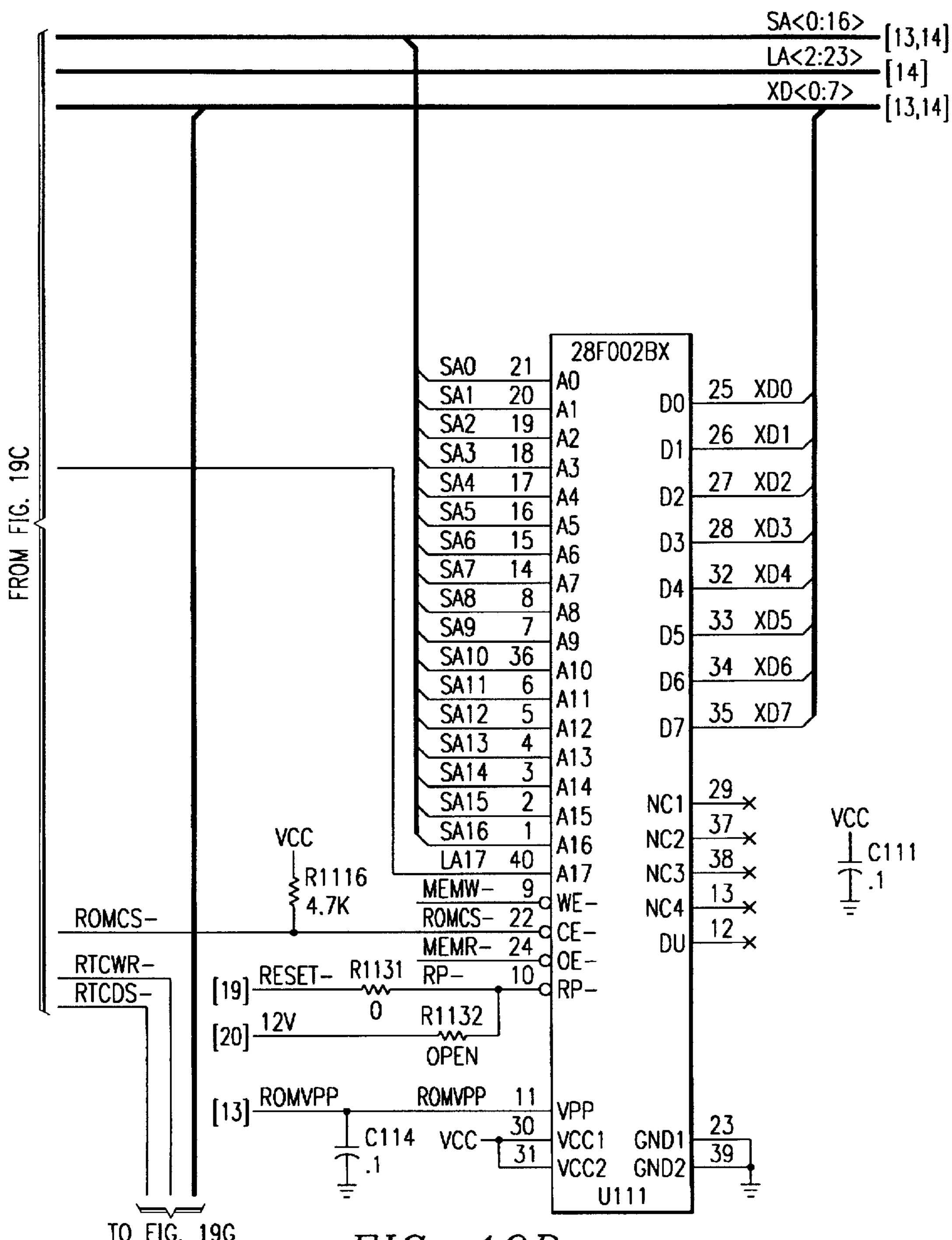

RP16
10K
RN112
16
VCC
R117 10K
R118 10K
R1128 10K
IORB-
[13,14]
SA0 SA1 SA2 SA3 SA4 SA5 SA6 SA7 SA8 SA9 SA10 SA11 SA12 SA13 SA14 SA15 SA16
SA<0:16>
A<2:23>
LA<2:23>
XD<0:7>
FROM FIG. 19B
TO FIG. 19D
VCC
ABT245PW
VCC
IOR-
A17 A18 A19 A20 A21 A22 A23
A1 A2 A3 A4 A5 A6 A7 A8
B1 B2 B3 B4 B5 B6 B7 B8
GND
U113
R112 10K
R111 10K
R110 10K
IORB-
LA17 LA18 LA19 LA20 LA21 LA22 LA23
R113 10K
R114 10K
R115 10K
R116 10K
ROMCS-
RTCWR-
RTCDS-

FIG. 19C



SA<0:16> [13,14]
LA<2:23> [14]
XD<0:7> [13,14]
FROM FIG. 19C
28F002BX
SA0 21 A0
SA1 20 A1
SA2 19 A2
SA3 18 A3
SA4 17 A4
SA5 16 A5
SA6 15 A6
SA7 14 A7
SA8 8 A8
SA9 7 A9
SA10 36 A10
SA11 6 A11
SA12 5 A12
SA13 4 A13
SA14 3 A14
SA15 2 A15
SA16 1 A16
LA17 40 A17
MEMW- 9 WE-
ROMCS- 22 CE-
MEMR- 24 OE-
RP- 10 RP-
D0 25 XD0
D1 26 XD1
D2 27 XD2
D3 28 XD3
D4 32 XD4
D5 33 XD5
D6 34 XD6
D7 35 XD7
NC1 29
NC2 37
NC3 38
NC4 13
DU 12
VCC
C111
.1
VCC
R1116
4.7K
ROMCS-
RTCWR-
RTCDS-
[19] RESET- R1131 0
[20] 12V R1132 OPEN
[13] ROMVPP ROMVPP 11 VPP
C114 .1
VCC 30 VCC1
31 VCC2
GND1 23
GND2 39
U111
TO FIG. 19G

FIG. 19D

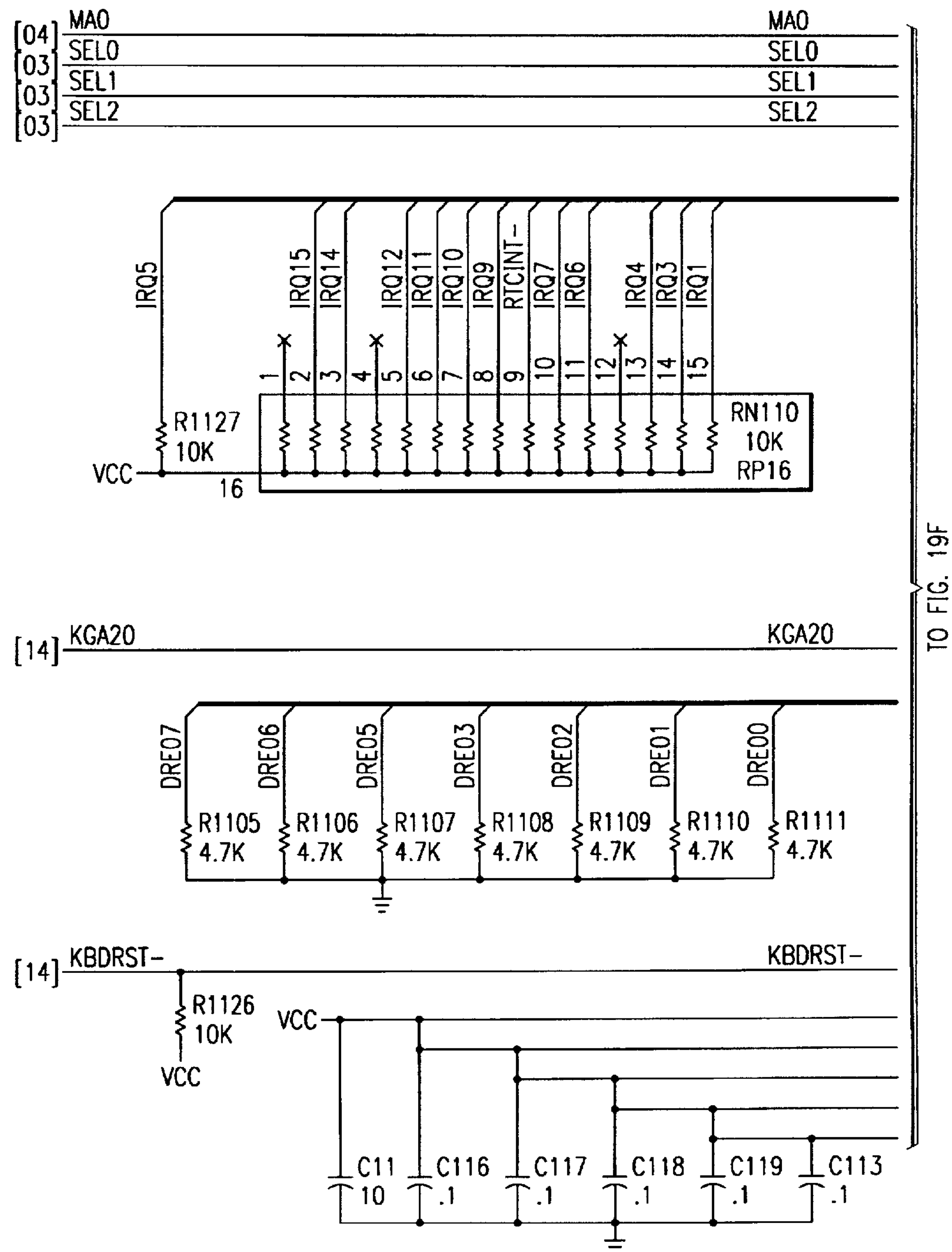

[04] MA0
[03] SEL0
[03] SEL1
[03] SEL2
MA0
SEL0
SEL1
SEL2
IRQ5
IRQ15
IRQ14
IRQ12
IRQ11
IRQ10
IRQ9
RTCINT-
IRQ7
IRQ6
IRQ4
IRQ3
IRQ1
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
R1127
10K
VCC
16
RN110
10K
RP16
TO FIG. 19F
[14] KGA20
KGA20
DRE07
DRE06
DRE05
DRE03
DRE02
DRE01
DRE00
R1105 4.7K
R1106 4.7K
R1107 4.7K
R1108 4.7K
R1109 4.7K
R1110 4.7K
R1111 4.7K
[14] KBDRST-
KBDRST-
R1126
10K
VCC
VCC
C11 10
C116 .1
C117 .1
C118 .1
C119 .1
C113 .1

FIG. 19E

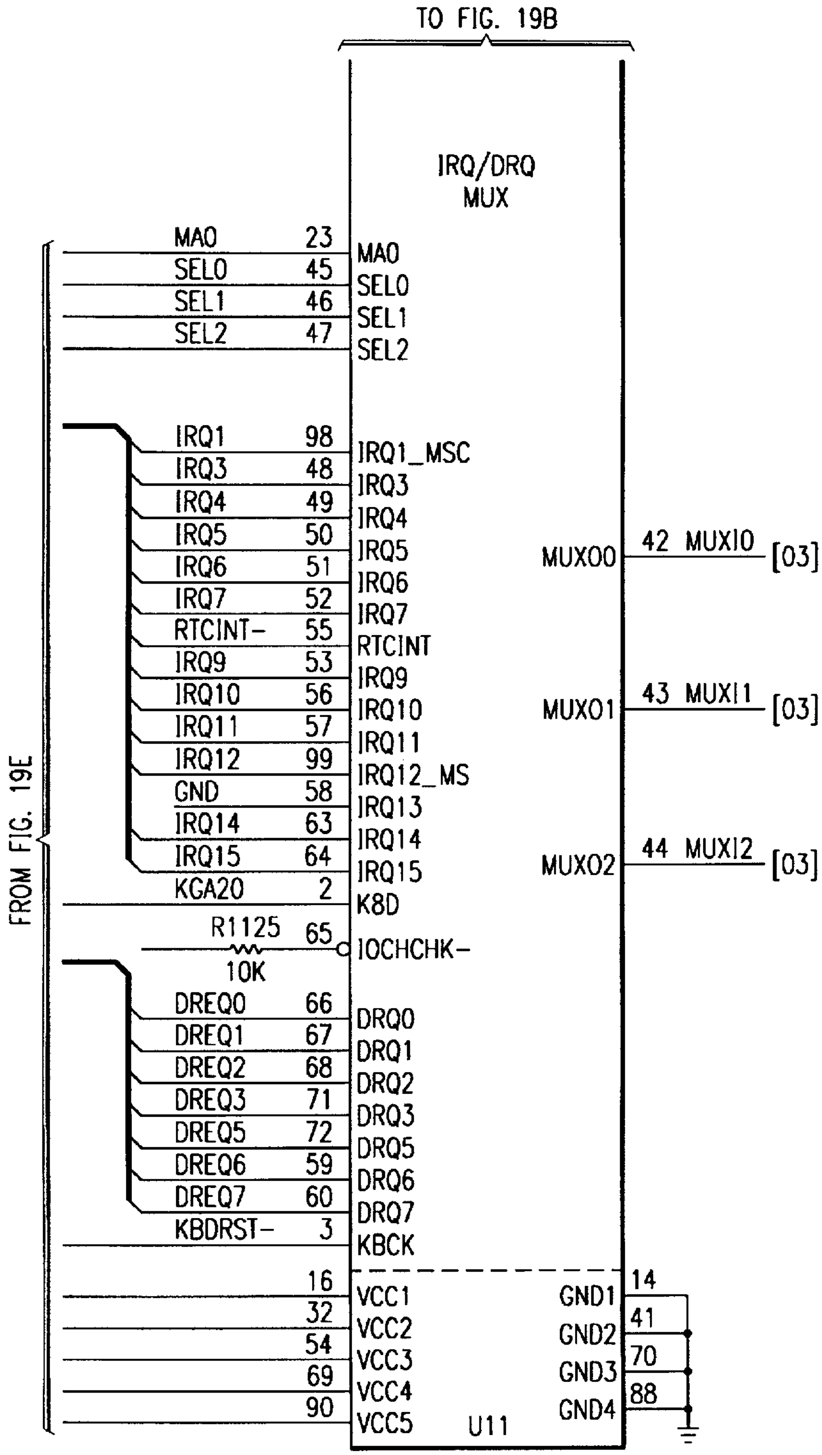

TO FIG. 19B
IRQ/DRQ
MUX
FROM FIG. 19E
MA0 23 MA0
SEL0 45 SEL0
SEL1 46 SEL1
SEL2 47 SEL2
IRQ1 98 IRQ1_MSC
IRQ3 48 IRQ3
IRQ4 49 IRQ4
IRQ5 50 IRQ5
IRQ6 51 IRQ6
IRQ7 52 IRQ7
RTCINT- 55 RTCINT
IRQ9 53 IRQ9
IRQ10 56 IRQ10
IRQ11 57 IRQ11
IRQ12 99 IRQ12_MS
GND 58 IRQ13
IRQ14 63 IRQ14
IRQ15 64 IRQ15
KGA20 2 K8D
R1125 10K 65 IOCHCHK-
DREQ0 66 DRQ0
DREQ1 67 DRQ1
DREQ2 68 DRQ2
DREQ3 71 DRQ3
DREQ5 72 DRQ5
DREQ6 59 DRQ6
DREQ7 60 DRQ7
KBDRST- 3 KBCK
16 VCC1
32 VCC2
54 VCC3
69 VCC4
90 VCC5
U11
MUXO0 42 MUXI0 [03]
MUXO1 43 MUXI1 [03]
MUXO2 44 MUXI2 [03]
GND1 14
GND2 41
GND3 70
GND4 88

FIG. 19F

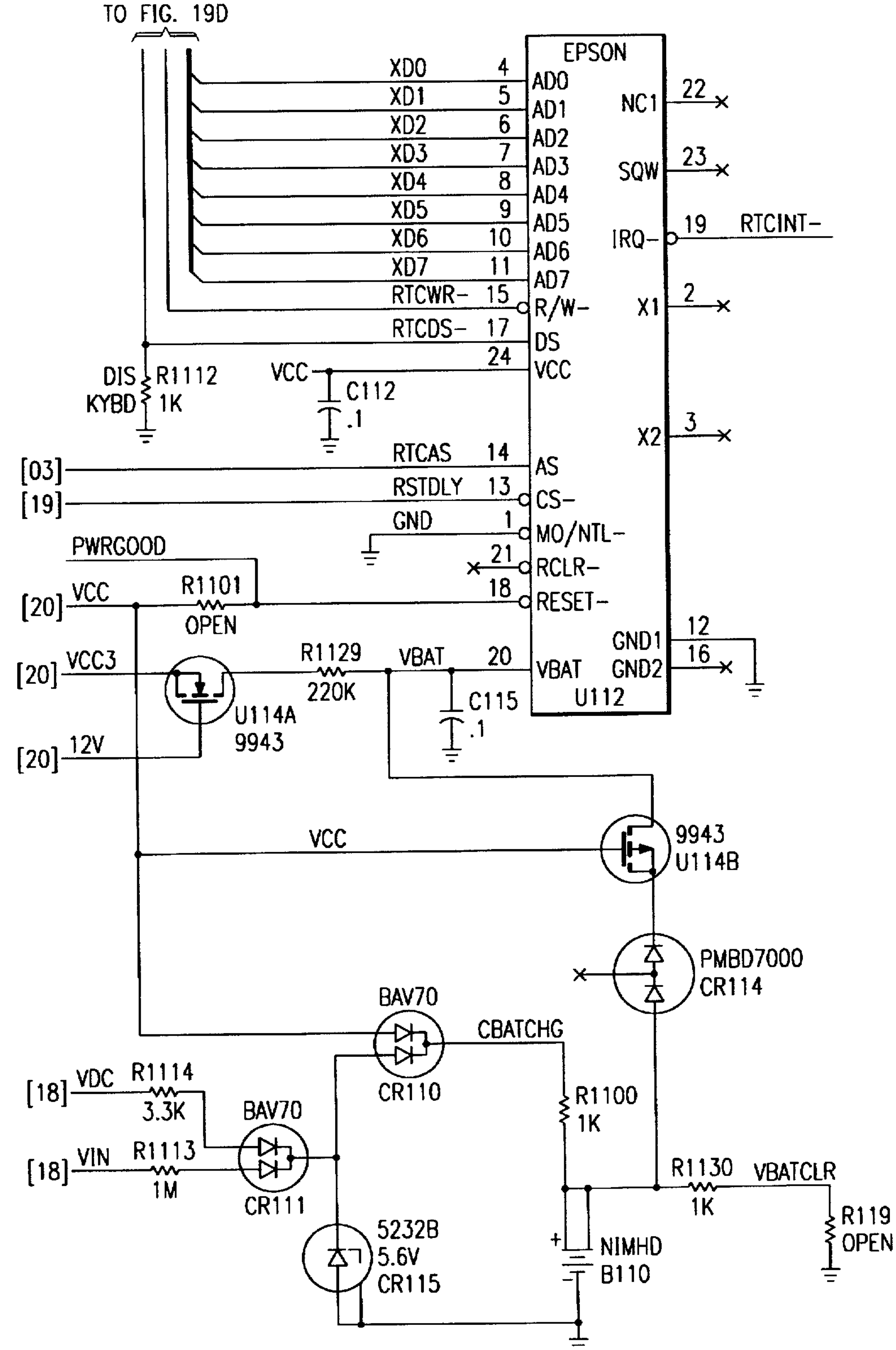
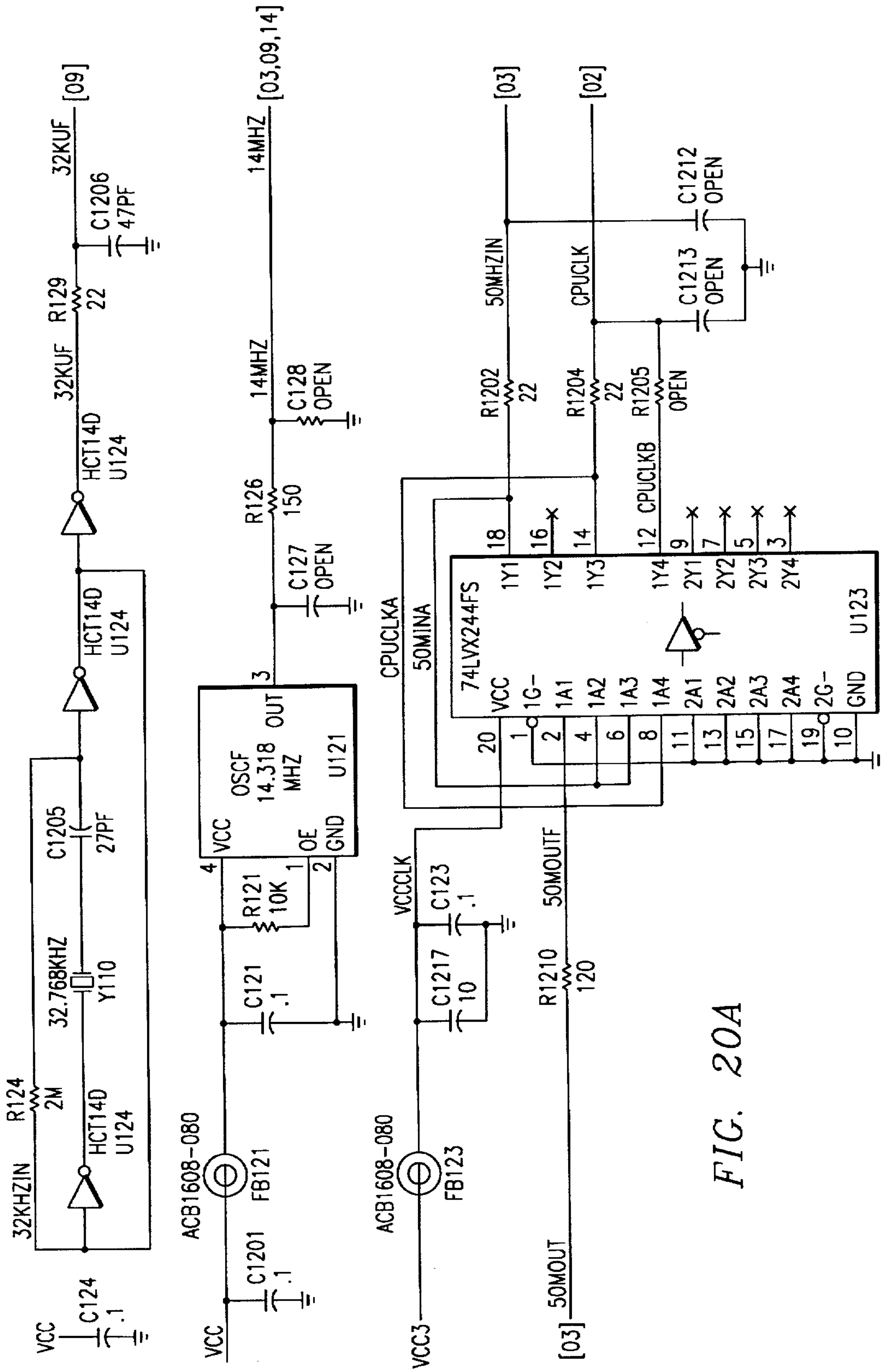
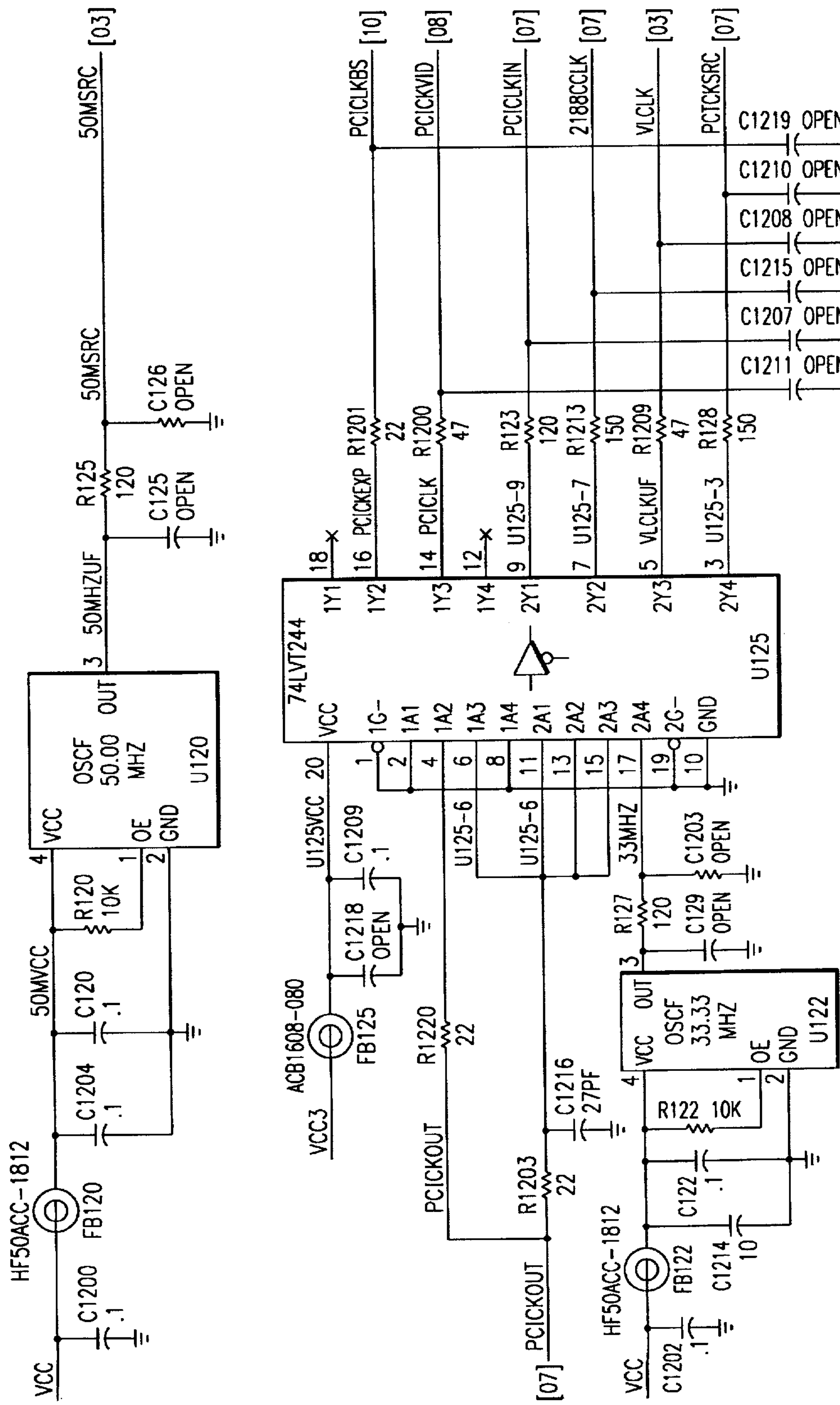

TO FIG. 19D
EPSON
XD0 4 AD0
XD1 5 AD1
XD2 6 AD2
XD3 7 AD3
XD4 8 AD4
XD5 9 AD5
XD6 10 AD6
XD7 11 AD7
RTCWR- 15 R/W-
RTCDS- 17 DS
VCC 24 VCC
NC1 22
SQW 23
IRQ- 19 RTCINT-
X1 2
X2 3
DIS KYBD R1112 1K
C112 .1
[03] RTCAS 14 AS
[19] RSTDLY 13 CS-
GND 1 MO/NTL-
21 RCLR-
PWRGOOD
[20] VCC R1101 OPEN 18 RESET-
GND1 12
GND2 16
[20] VCC3 R1129 220K VBAT 20 VBAT
U112
C115 .1
U114A 9943
[20] 12V
VCC
9943 U114B
PMBD7000 CR114
BAV70 CR110
CBATCHG
[18] VDC R1114 3.3K
BAV70
[18] VIN R1113 1M
CR111
R1100 1K
R1130 1K VBATCLR
R119 OPEN
5232B 5.6V CR115
NIMHD B110

FIG. 19G



VCC
C124
.1
32KHZIN
R124
2M
HCT14D
U124
32.768KHZ
Y110
C1205
27PF
HCT14D
U124
HCT14D
U124
32KUF
R129
22
C1206
47PF
32KUF
[09]
VCC
C1201
.1
ACB1608-080
FB121
C121
.1
R121
10K
VCC
OE
GND
OSCF
14.318
MHZ
OUT
U121
C127
OPEN
R126
150
C128
OPEN
14MHZ
14MHZ
[03,09,14]
VCC3
ACB1608-080
FB123
C1217
10
VCCCLK
C123
.1
CPUCLKA
50MINA
74LVX244FS
U123
VCC
1G-
1A1
1A2
1A3
1A4
2A1
2A2
2A3
2A4
2G-
GND
1Y1
1Y2
1Y3
1Y4
2Y1
2Y2
2Y3
2Y4
CPUCLKB
R1202
22
R1204
22
R1205
OPEN
50MHZIN
CPUCLK
C1213
OPEN
C1212
OPEN
[03]
[02]
[03]
50MOUT
R1210
120
50MOUTF

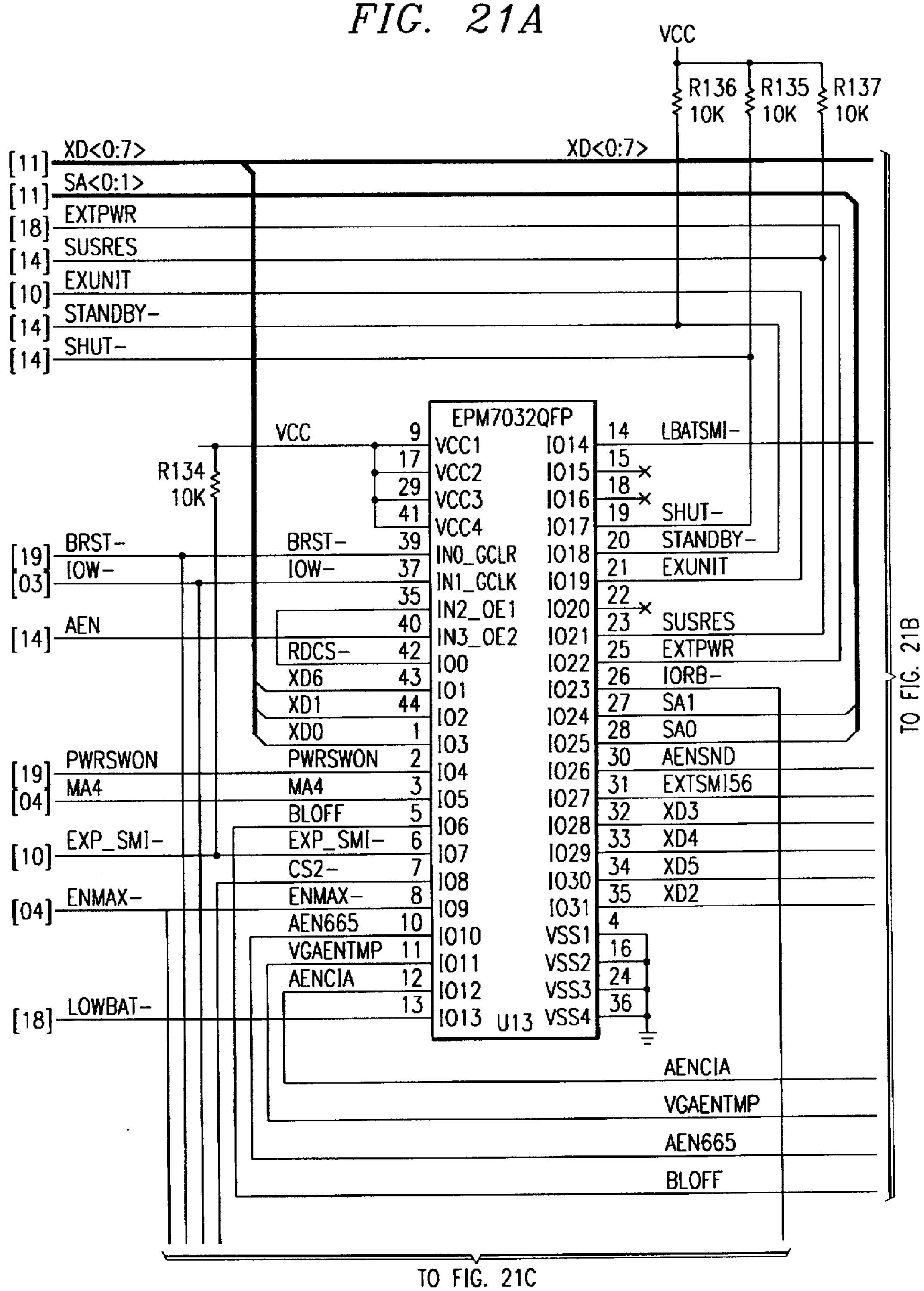
FIG. 20A



VCC
C1200 .1
HF50ACC-1812
FB120
C1204 .1
50MVCC
C120 .1
R120 10K
OSCF 50.00 MHZ
U120
VCC
OE
GND
OUT
50MHZUF
R125 120
C125 OPEN
50MSRC
C126 OPEN
50MSRC [03]
VCC3
ACB1608-080
FB125
C1218 OPEN
C1209 .1
U125VCC
74LVT244
U125
1G-
1A1
1A2
1A3
1A4
2A1
2A2
2A3
2A4
2G-
GND
1Y1
1Y2
1Y3
1Y4
2Y1
2Y2
2Y3
2Y4
PCICKEXP
PCICLK
U125-9
U125-7
VLCLKUF
U125-3
U125-6
33MHZ
R1201 22
R1200 47
R123 120
R1213 150
R1209 47
R128 150
PCICLKBS [10]
PCICKVID [08]
PCICLKIN [07]
218BCCLK [07]
VLCLK [03]
PCICKSRC [07]
C1219 OPEN
C1210 OPEN
C1208 OPEN
C1215 OPEN
C1207 OPEN
C1211 OPEN
R1220 22
PCICKOUT
R1203 22
C1216 27PF
[07] PCICKOUT
C1203 OPEN
R127 120
C129 OPEN
OSCF 33.33 MHZ
U122
R122 10K
C122 .1
C1214 10
HF50ACC-1812
FB122
C1202 .1

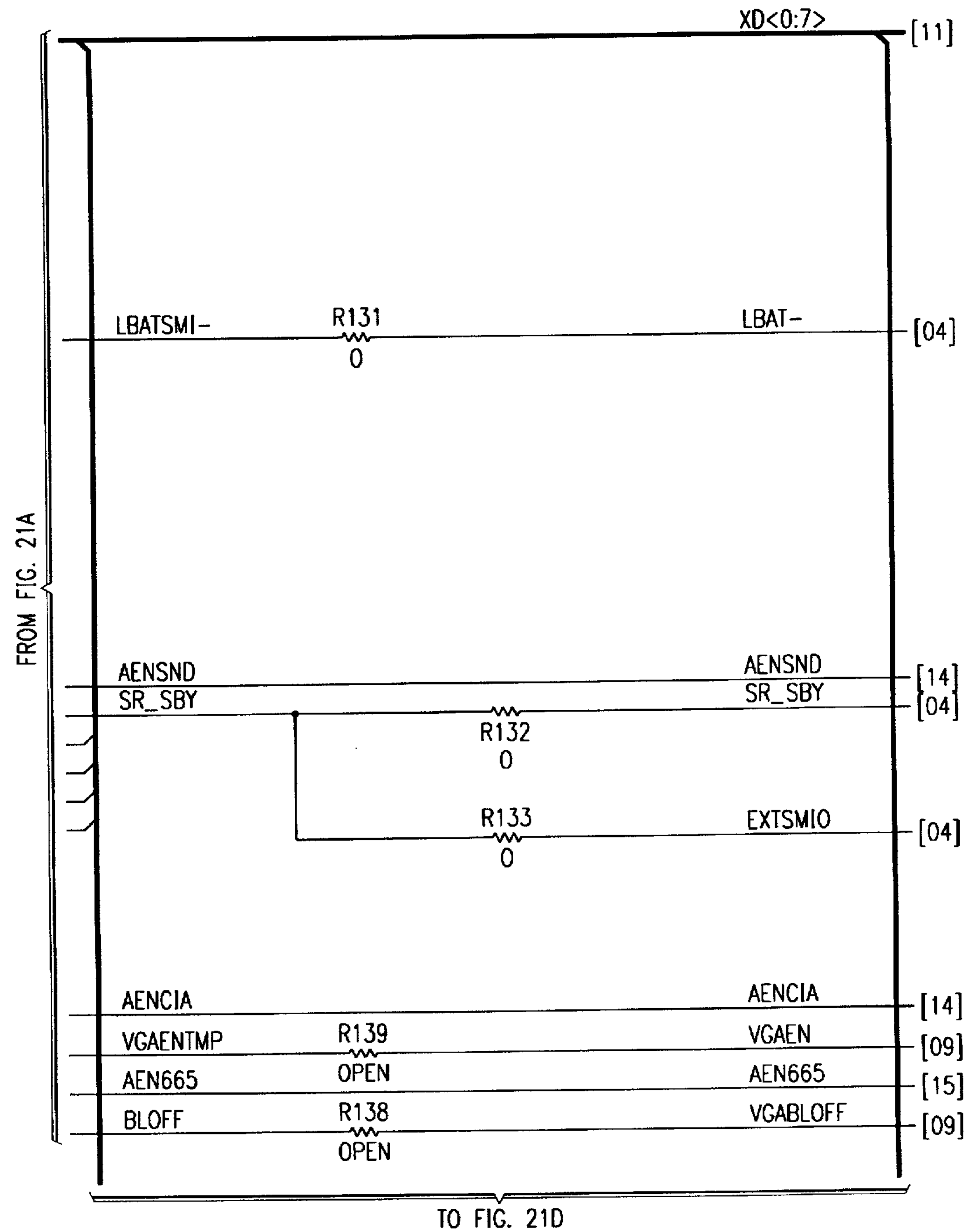
FIG. 20B

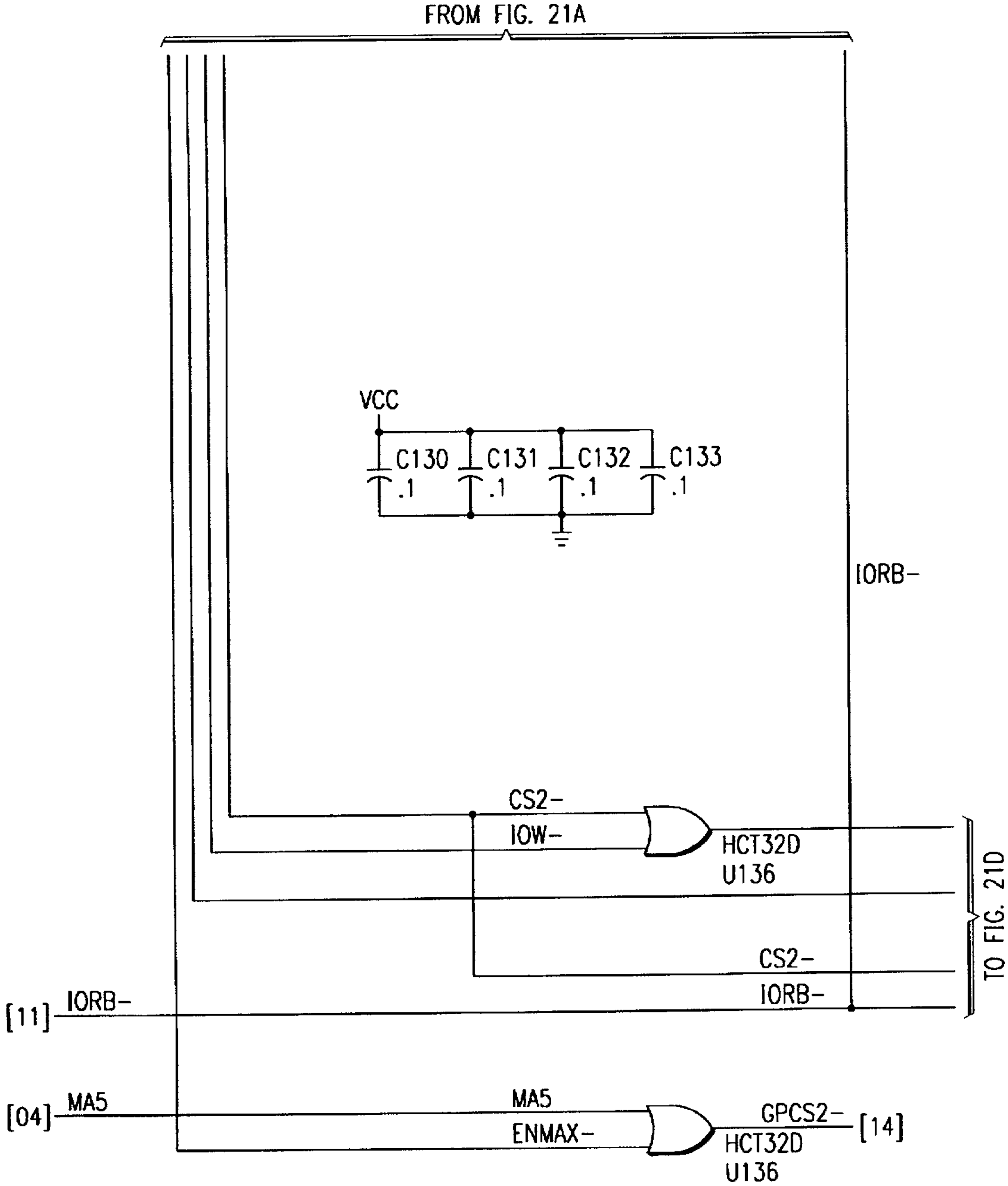
FIG. 21A



VCC
R136 10K
R135 10K
R137 10K
XD<0:7>
SA<0:1>
EXTPWR
SUSRES
EXUNIT
STANDBY-
SHUT-
[11] [11] [18] [14] [10] [14] [14]
EPM7032QFP
U13
VCC1 VCC2 VCC3 VCC4
IN0_GCLR IN1_GCLK IN2_OE1 IN3_OE2
IO0 IO1 IO2 IO3 IO4 IO5 IO6 IO7 IO8 IO9 IO10 IO11 IO12 IO13
IO14 IO15 IO16 IO17 IO18 IO19 IO20 IO21 IO22 IO23 IO24 IO25 IO26 IO27 IO28 IO29 IO30 IO31
VSS1 VSS2 VSS3 VSS4
R134 10K
BRST- IOW- AEN RDCS- XD6 XD1 XD0 PWRSWON MA4 BLOFF EXP_SMI- CS2- ENMAX- AEN665 VGAENTMP AENCIA LOWBAT-
[19] [03] [14] [19] [04] [10] [04] [18]
LBATSMI- SHUT- STANDBY- EXUNIT SUSRES EXTPWR IORB- SA1 SA0 AENSND EXTSMI56 XD3 XD4 XD5 XD2
AENCIA VGAENTMP AEN665 BLOFF
TO FIG. 21B
TO FIG. 21C

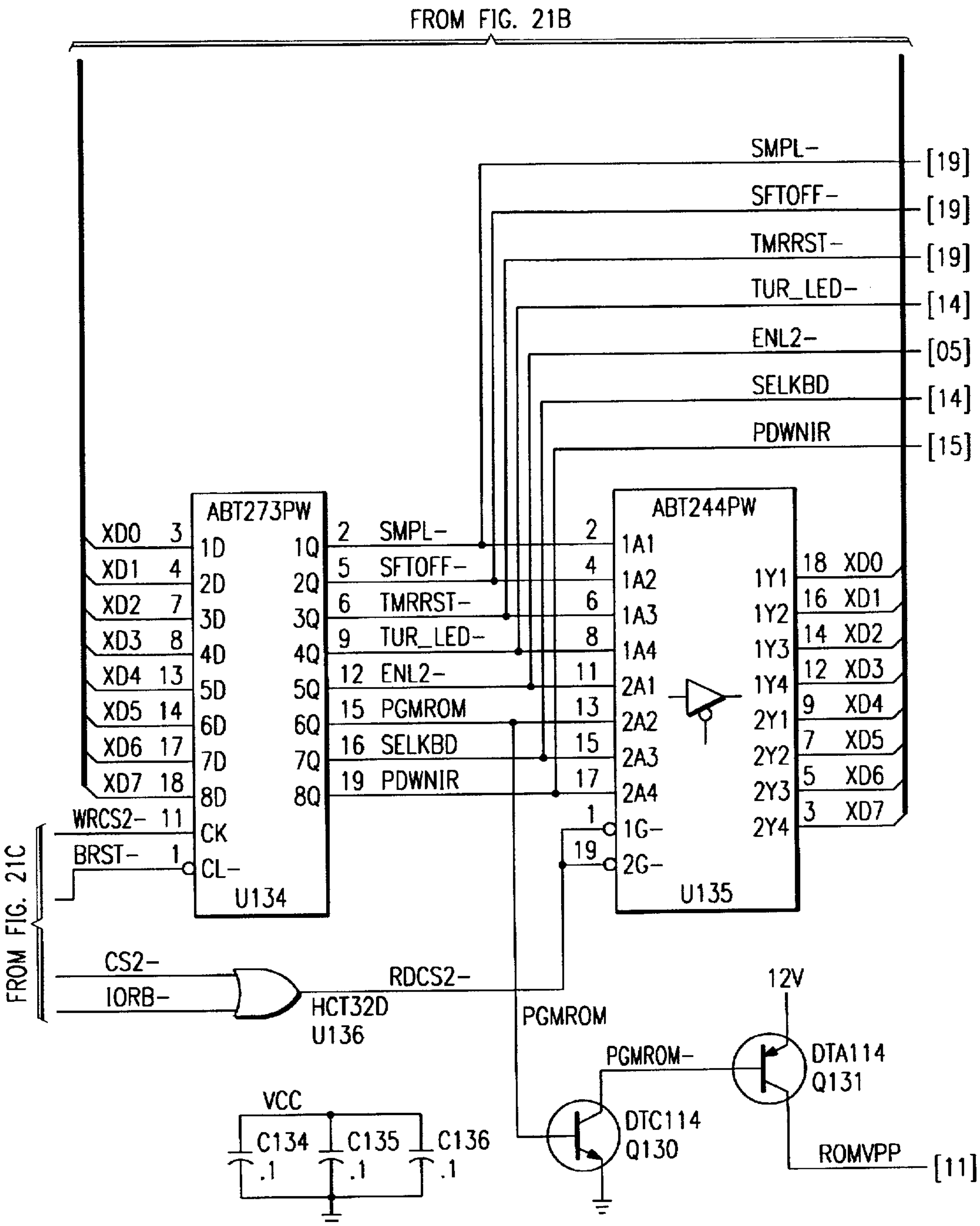
FIG. 21B


XD<0:7>
[11]
FROM FIG. 21A
LBATSMI-
R131
0
LBAT-
[04]
AENSND
SR_SBY
AENSND
SR_SBY
[14]
[04]
R132
0
R133
0
EXTSMIO
[04]
AENCIA
AENCIA
[14]
VGAENTMP
R139
OPEN
VGAEN
[09]
AEN665
AEN665
[15]
BLOFF
R138
OPEN
VGABLOFF
[09]
TO FIG. 21D



FROM FIG. 21A
VCC
C130
.1
C131
.1
C132
.1
C133
.1
IORB-
CS2-
IOW-
HCT32D
U136
TO FIG. 21D
CS2-
IORB-
[11] IORB-
[04] MA5
MA5
ENMAX-
GPCS2-
HCT32D
U136
[14]

FIG. 21C



FROM FIG. 21B
SMPL- [19]
SFTOFF- [19]
TMRRST- [19]
TUR_LED- [14]
ENL2- [05]
SELKBD [14]
PDWNIR [15]
ABT273PW
XD0 3 1D 1Q 2 SMPL-
XD1 4 2D 2Q 5 SFTOFF-
XD2 7 3D 3Q 6 TMRRST-
XD3 8 4D 4Q 9 TUR_LED-
XD4 13 5D 5Q 12 ENL2-
XD5 14 6D 6Q 15 PGMROM
XD6 17 7D 7Q 16 SELKBD
XD7 18 8D 8Q 19 PDWNIR
WRCS2- 11 CK
BRST- 1 CL-
U134
ABT244PW
2 1A1
4 1A2
6 1A3
8 1A4
11 2A1
13 2A2
15 2A3
17 2A4
1 1G-
19 2G-
1Y1 18 XD0
1Y2 16 XD1
1Y3 14 XD2
1Y4 12 XD3
2Y1 9 XD4
2Y2 7 XD5
2Y3 5 XD6
2Y4 3 XD7
U135
FROM FIG. 21C
CS2-
IORB-
RDCS2-
HCT32D
U136
PGMROM
12V
PGMROM-
DTA114
Q131
DTC114
Q130
ROMVPP [11]
VCC
C134 .1
C135 .1
C136 .1

FIG. 21D

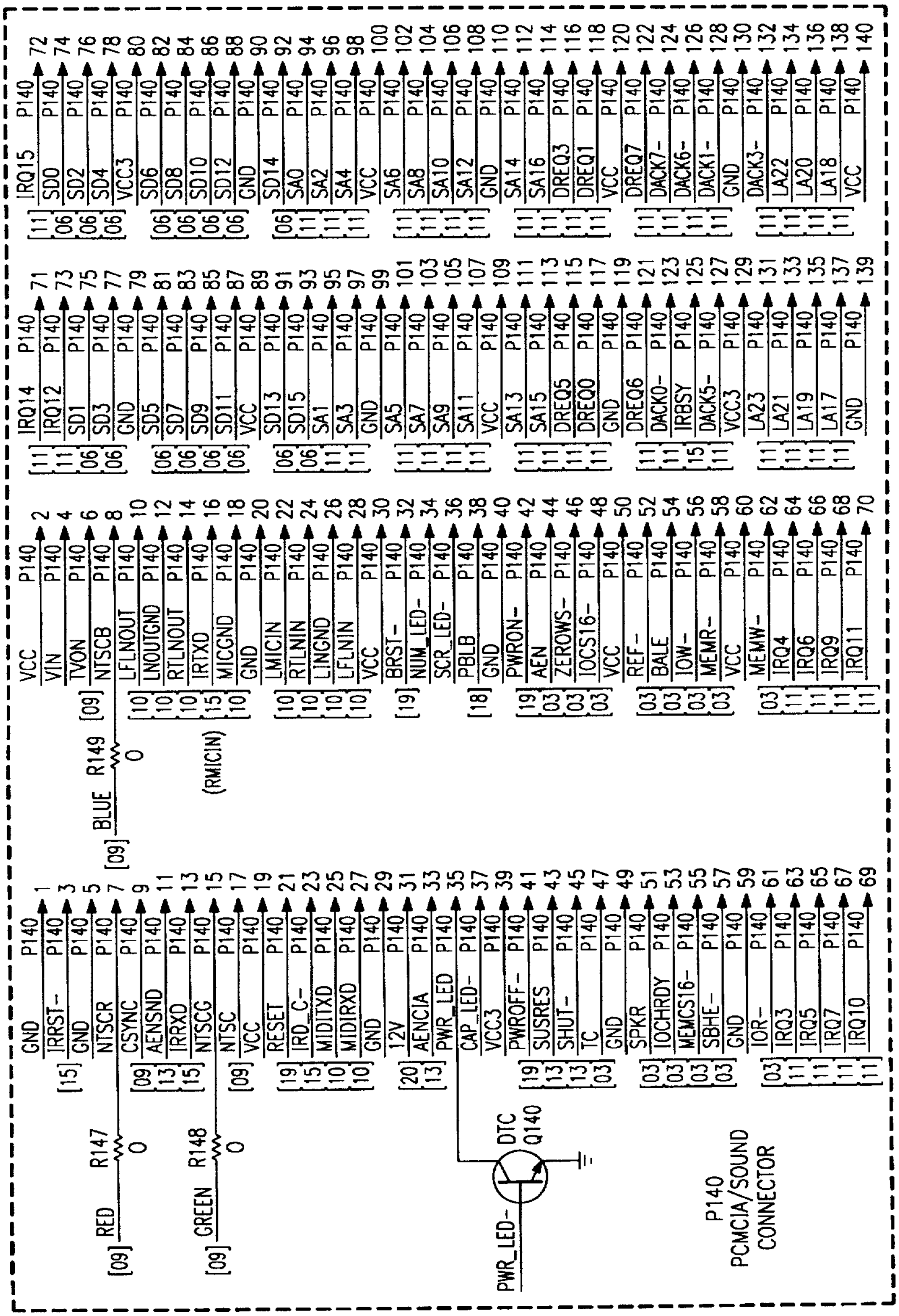
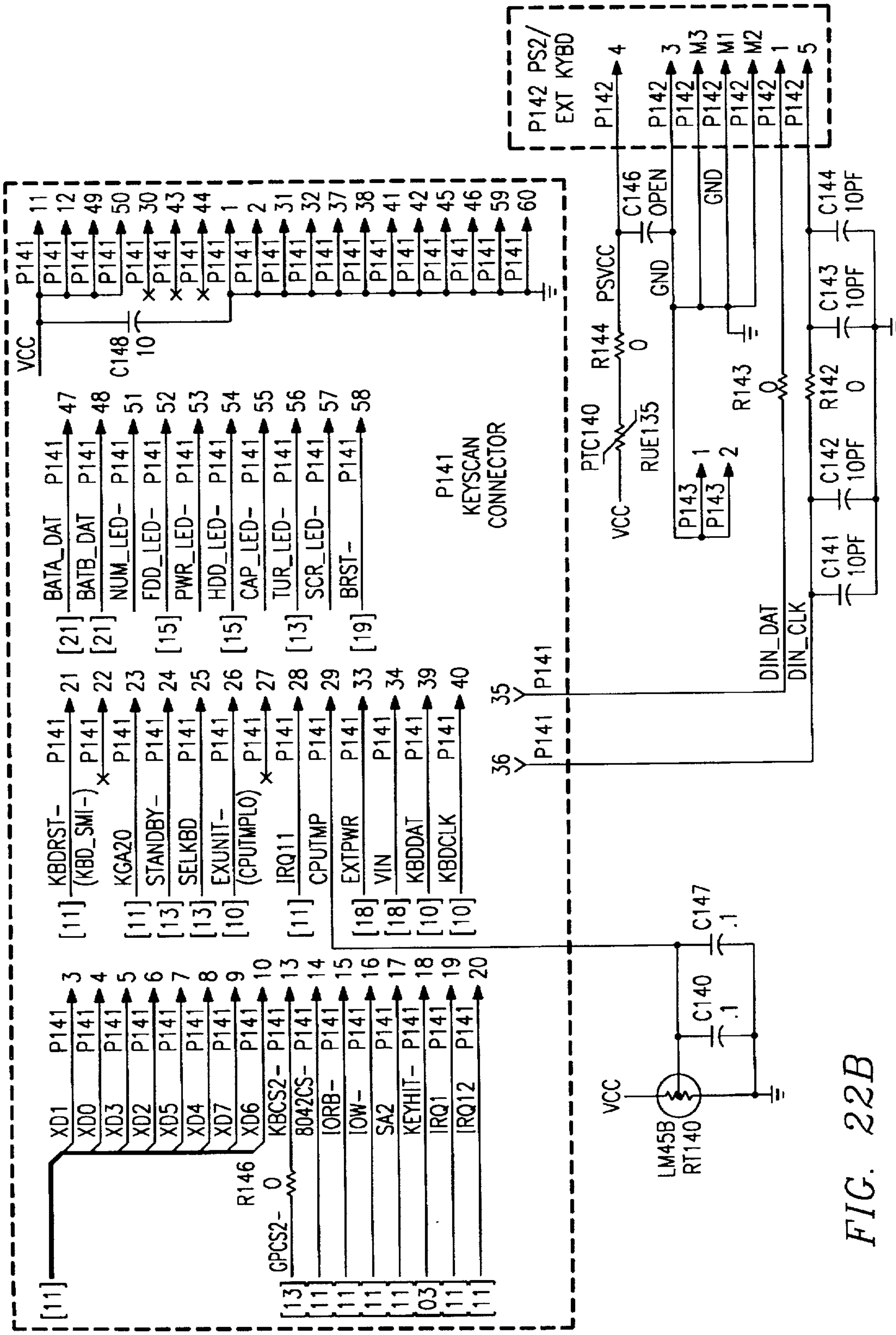
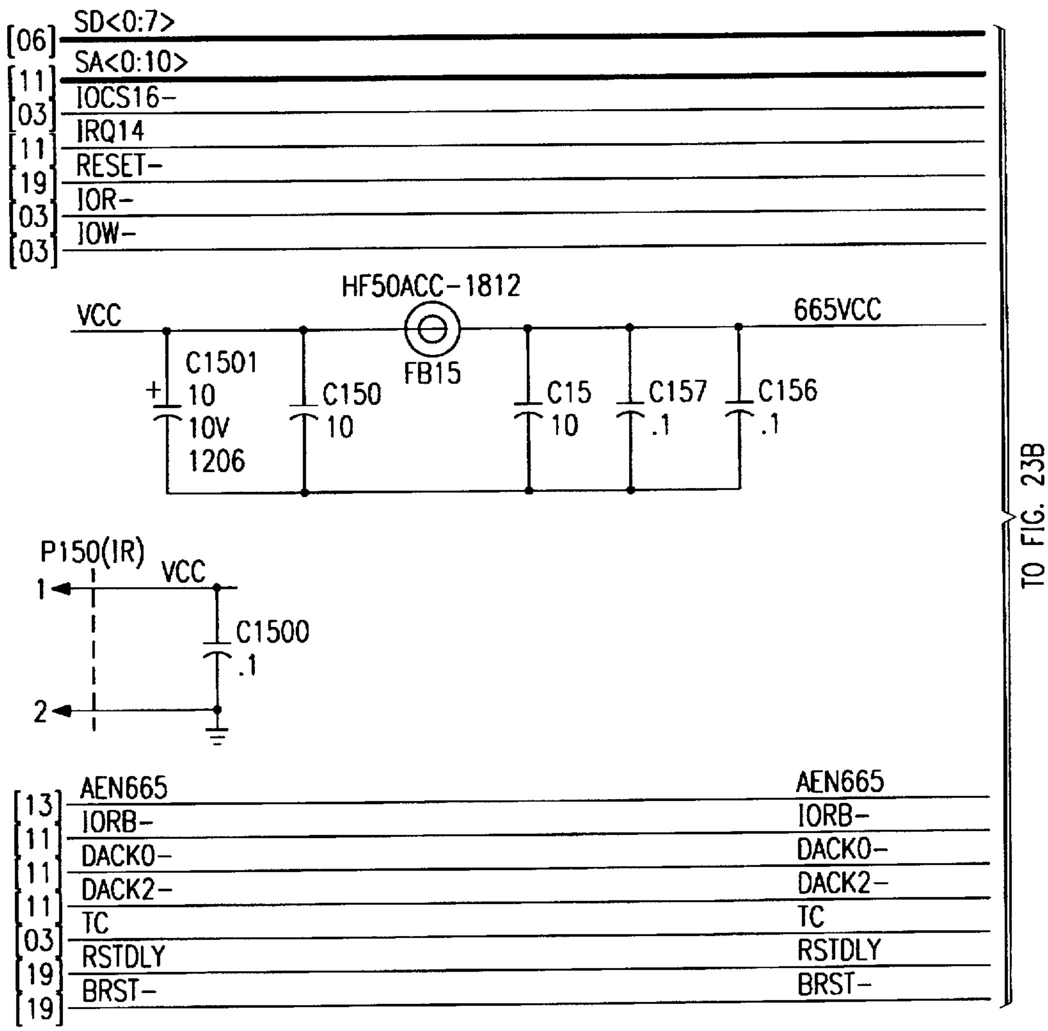

GND P140 1
IRRST- P140 3
GND P140 5
NTSCR P140 7
CSYNC P140 9
AENSND P140 11
IRRXD P140 13
NTSCG P140 15
NTSC P140 17
VCC P140 19
RESET P140 21
IRD_C- P140 23
MIDITXD P140 25
MIDIRXD P140 27
GND P140 29
12V P140 31
AENCIA P140 33
PWR_LED P140 35
CAP_LED- P140 37
VCC3 P140 39
PWROFF- P140 41
SUSRES P140 43
SHUT- P140 45
TC P140 47
GND P140 49
SPKR P140 51
IOCHRDY P140 53
MEMCS16- P140 55
SBHE- P140 57
GND P140 59
IOR- P140 61
IRQ3 P140 63
IRQ5 P140 65
IRQ7 P140 67
IRQ10 P140 69
VCC P140 2
VIN P140 4
TVON P140 6
NTSCB P140 8
LFLNOUT P140 10
LNOUTGND P140 12
RTLNOUT P140 14
IRTXD P140 16
MICGND P140 18
GND P140 20
LMICIN P140 22
RTLNIN P140 24
LINGND P140 26
LFLNIN P140 28
VCC P140 30
BRST- P140 32
NUM_LED- P140 34
SCR_LED- P140 36
PBLB P140 38
GND P140 40
PWRON- P140 42
AEN P140 44
ZEROWS- P140 46
IOCS16- P140 48
VCC P140 50
REF- P140 52
BALE P140 54
IOW- P140 56
MEMR- P140 58
VCC P140 60
MEMW- P140 62
IRQ4 P140 64
IRQ6 P140 66
IRQ9 P140 68
IRQ11 P140 70
IRQ14 P140 71
IRQ12 P140 73
SD1 P140 75
SD3 P140 77
GND P140 79
SD5 P140 81
SD7 P140 83
SD9 P140 85
SD11 P140 87
VCC P140 89
SD13 P140 91
SD15 P140 93
SA1 P140 95
SA3 P140 97
GND P140 99
SA5 P140 101
SA7 P140 103
SA9 P140 105
SA11 P140 107
VCC P140 109
SA13 P140 111
SA15 P140 113
DREQ5 P140 115
DREQ0 P140 117
GND P140 119
DREQ6 P140 121
DACK0- P140 123
IRBSY P140 125
DACK5- P140 127
VCC3 P140 129
LA23 P140 131
LA21 P140 133
LA19 P140 135
LA17 P140 137
GND P140 139
IRQ15 P140 72
SD0 P140 74
SD2 P140 76
SD4 P140 78
VCC3 P140 80
SD6 P140 82
SD8 P140 84
SD10 P140 86
SD12 P140 88
GND P140 90
SD14 P140 92
SA0 P140 94
SA2 P140 96
SA4 P140 98
VCC P140 100
SA6 P140 102
SA8 P140 104
SA10 P140 106
SA12 P140 108
GND P140 110
SA14 P140 112
SA16 P140 114
DREQ3 P140 116
DREQ1 P140 118
VCC P140 120
DREQ7 P140 122
DACK7- P140 124
DACK6- P140 126
DACK1- P140 128
GND P140 130
DACK3- P140 132
LA22 P140 134
LA20 P140 136
LA18 P140 138
VCC P140 140
[09] RED R147 0
[09] GREEN R148 0
[09] BLUE R149 0
(RMICIN)
PWR_LED-
DTC Q140
P140
PCMCIA/SOUND
CONNECTOR

FIG. 22A



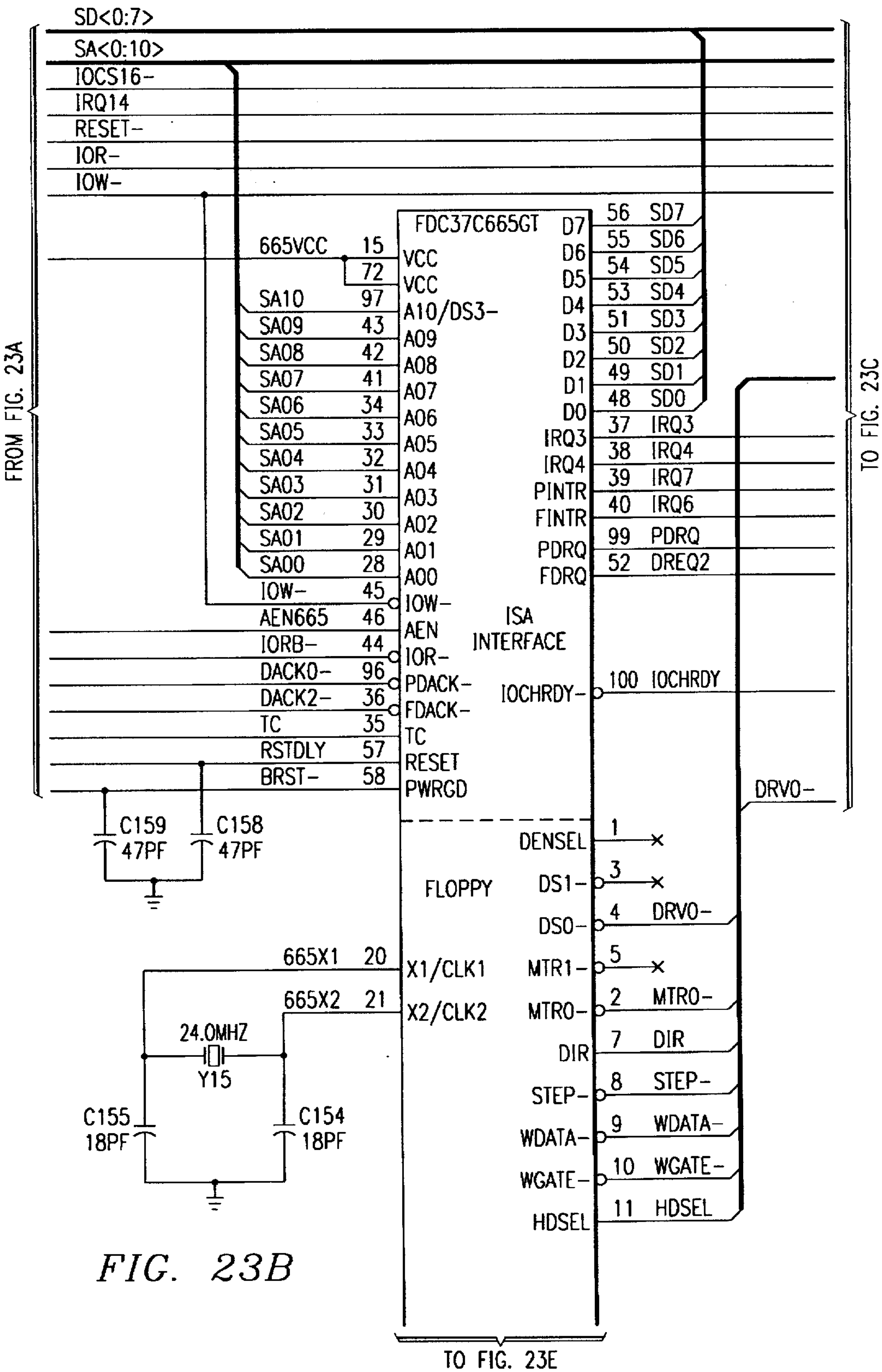
P142 PS2/
EXT KYBD
P141
KEYSCAN
CONNECTOR
VCC
C148
10
PSVCC
GND
C146
OPEN
R144
0
PTC140
RUE135
P143
R143
0
R142
0
C141
10PF
C142
10PF
C143
10PF
C144
10PF
DIN_DAT
DIN_CLK
BATA_DAT
BATB_DAT
NUM_LED-
FDD_LED-
PWR_LED-
HDD_LED-
CAP_LED-
TUR_LED-
SCR_LED-
BRST-
KBDRST-
(KBD_SMI-)
KGA20
STANDBY-
SELKBD
EXUNIT-
(CPUTMPLO)
IRQ11
CPUTMP
EXTPWR
VIN
KBDDAT
KBDCLK
XD1
XD0
XD3
XD2
XD5
XD4
XD7
XD6
KBCS2-
8042CS-
IORB-
IOW-
SA2
KEYHIT-
IRQ1
IRQ12
GPCS2-
R146
0
C147
.1
C140
.1
LM45B
RT140

FIG. 22B



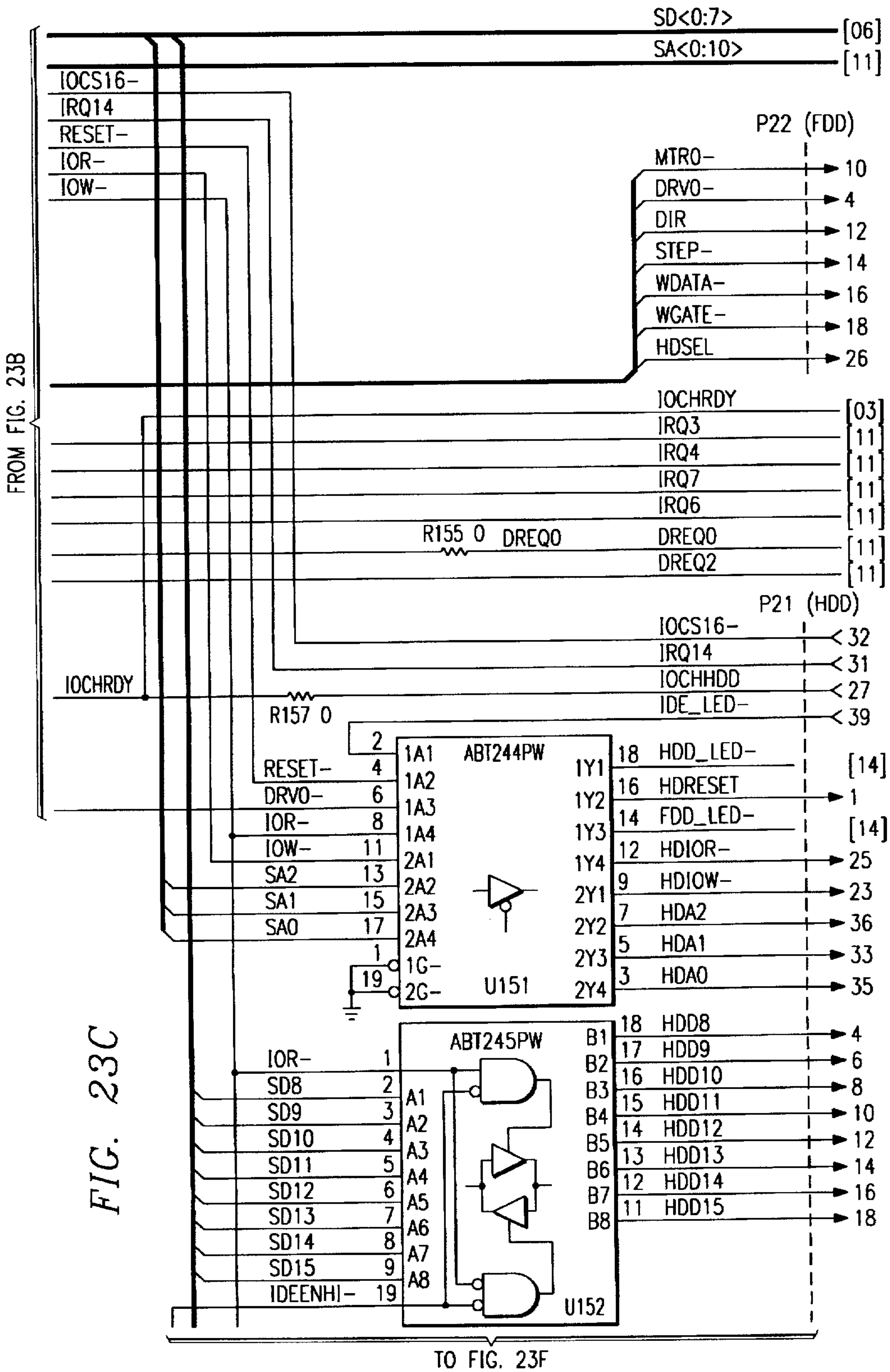
[06] SD<0:7>
[11] SA<0:10>
[03] IOCS16-
[11] IRQ14
[19] RESET-
[03] IOR-
[03] IOW-
HF50ACC-1812
VCC
665VCC
FB15
C1501
10
10V
1206
C150
10
C15
10
C157
.1
C156
.1
P150(IR)
VCC
1
2
C1500
.1
[13] AEN665
[11] IORB-
[11] DACK0-
[11] DACK2-
[03] TC
[19] RSTDLY
[19] BRST-
AEN665
IORB-
DACK0-
DACK2-
TC
RSTDLY
BRST-
TO FIG. 23B

FIG. 23A



SD<0:7>
SA<0:10>
IOCS16-
IRQ14
RESET-
IOR-
IOW-
FROM FIG. 23A
FDC37C665GT
665VCC
VCC
VCC
A10/DS3-
SA10
SA09
SA08
SA07
SA06
SA05
SA04
SA03
SA02
SA01
SA00
IOW-
AEN665
IORB-
DACK0-
DACK2-
TC
RSTDLY
BRST-
ISA INTERFACE
IOCHRDY-
100 IOCHRDY
56 SD7
55 SD6
54 SD5
53 SD4
51 SD3
50 SD2
49 SD1
48 SD0
37 IRQ3
38 IRQ4
39 IRQ7
40 IRQ6
99 PDRQ
52 DREQ2
PINTR
FINTR
PDRQ
FDRQ
TO FIG. 23C
DRV0-
C159 47PF
C158 47PF
FLOPPY
DENSEL
DS1-
DS0-
MTR1-
MTR0-
DIR
STEP-
WDATA-
WGATE-
HDSEL
665X1 20 X1/CLK1
665X2 21 X2/CLK2
24.0MHZ
Y15
C155 18PF
C154 18PF
4 DRV0-
2 MTR0-
7 DIR
8 STEP-
9 WDATA-
10 WGATE-
11 HDSEL
TO FIG. 23E

FIG. 23B



SD<0:7>
[06]
SA<0:10>
[11]
IOCS16-
IRQ14
RESET-
IOR-
IOW-
P22 (FDD)
MTRO-
DRVO-
DIR
STEP-
WDATA-
WGATE-
HDSEL
FROM FIG. 23B
IOCHRDY
[03]
IRQ3
IRQ4
IRQ7
IRQ6
R155 0
DREQ0
DREQ2
P21 (HDD)
IOCHHDD
IDE_LED-
R157 0
ABT244PW
U151
HDD_LED-
HDRESET
FDD_LED-
HDIOR-
HDIOW-
HDA2
HDA1
HDA0
[14]
SA2
SA1
SA0
ABT245PW
U152
SD8
SD9
SD10
SD11
SD12
SD13
SD14
SD15
IDEENHI-
HDD8
HDD9
HDD10
HDD11
HDD12
HDD13
HDD14
HDD15
TO FIG. 23F

FIG. 23C

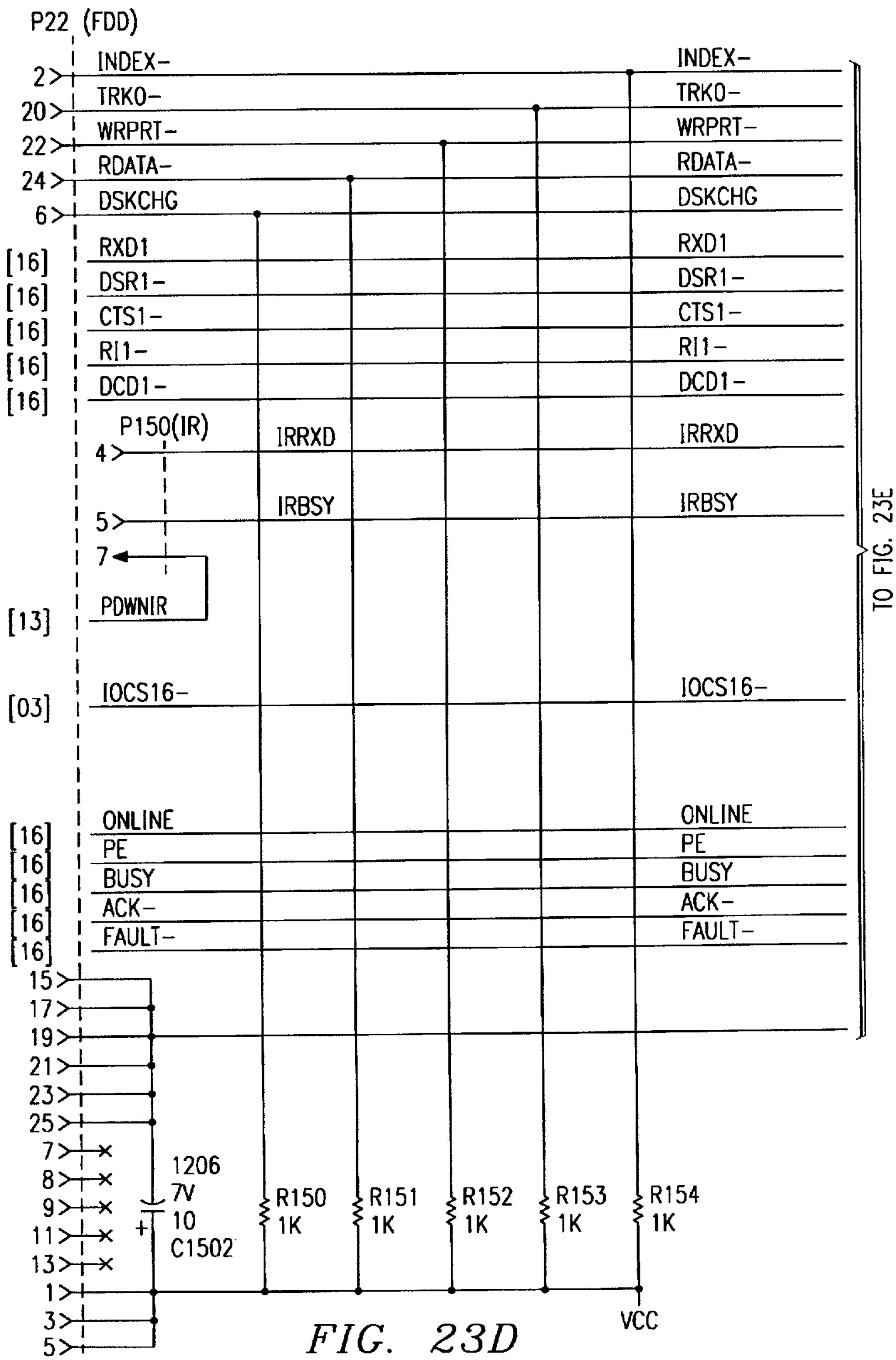
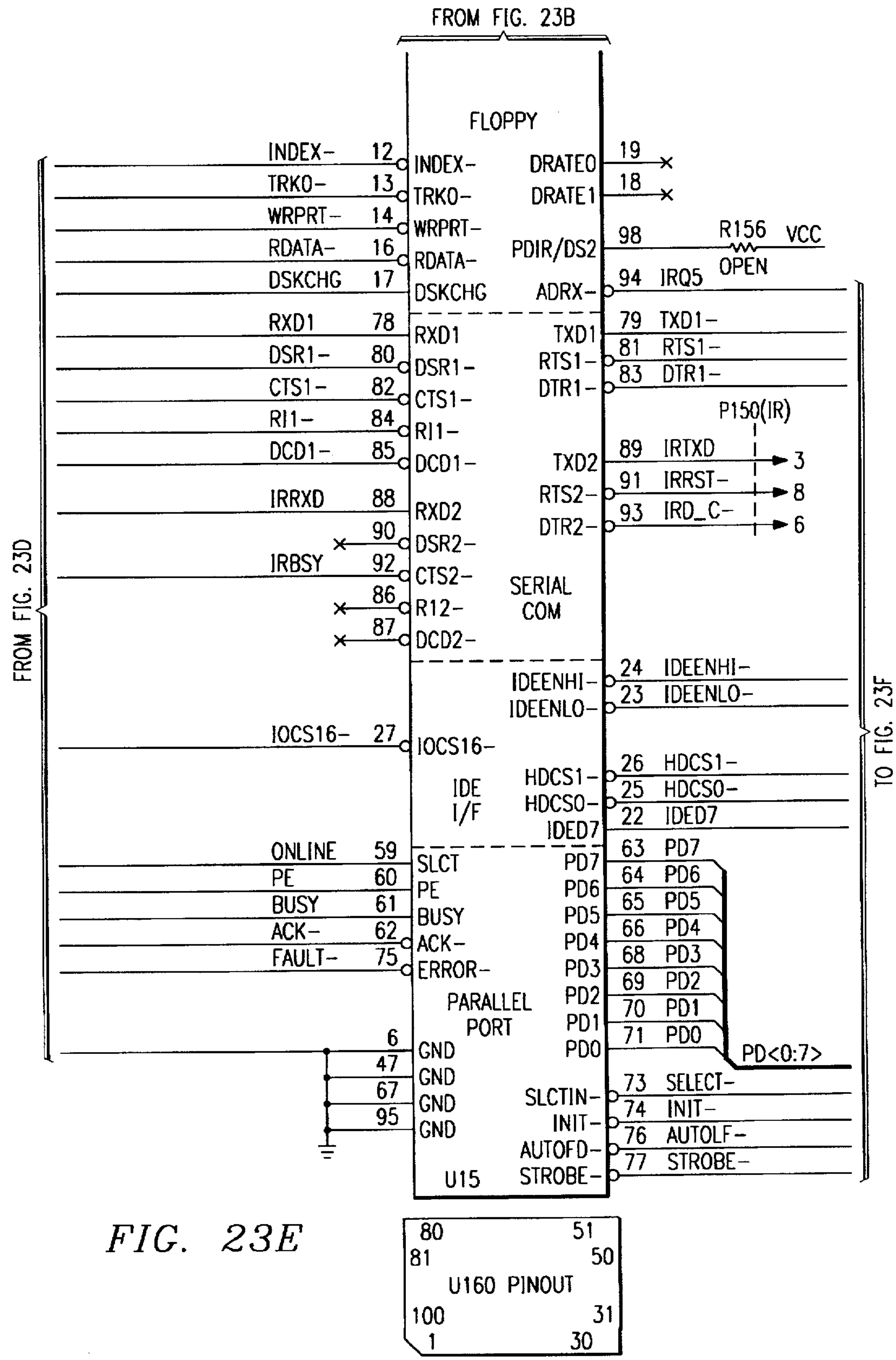
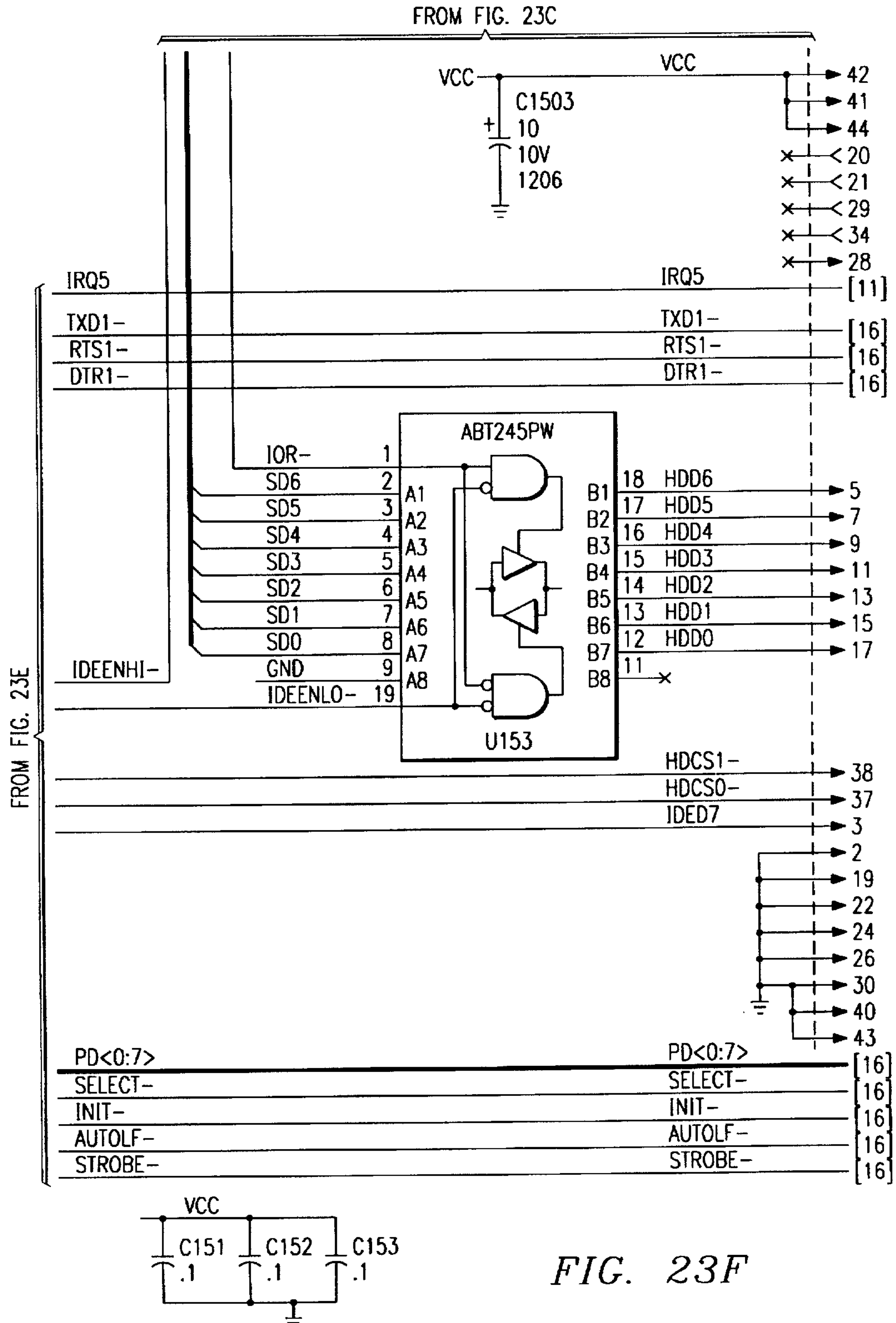

P22 (FDD)
INDEX-
TRK0-
WRPRT-
RDATA-
DSKCHG
RXD1
DSR1-
CTS1-
RI1-
DCD1-
P150(IR)
IRRXD
IRBSY
PDWNIR
IOCS16-
ONLINE
PE
BUSY
ACK-
FAULT-
1206
7V
10
C1502
R150 1K
R151 1K
R152 1K
R153 1K
R154 1K
VCC
TO FIG. 23E

FIG. 23D



FROM FIG. 23B
FLOPPY
INDEX- 12 INDEX-
TRK0- 13 TRK0-
WRPRT- 14 WRPRT-
RDATA- 16 RDATA-
DSKCHG 17 DSKCHG
DRATE0 19
DRATE1 18
PDIR/DS2 98 R156 VCC
OPEN
ADRX- 94 IRQ5
RXD1 78 RXD1
DSR1- 80 DSR1-
CTS1- 82 CTS1-
RI1- 84 RI1-
DCD1- 85 DCD1-
IRRXD 88 RXD2
90 DSR2-
IRBSY 92 CTS2-
86 RI2-
87 DCD2-
TXD1 79 TXD1-
RTS1- 81 RTS1-
DTR1- 83 DTR1-
P150(IR)
TXD2 89 IRTXD 3
RTS2- 91 IRRST- 8
DTR2- 93 IRD_C- 6
SERIAL COM
FROM FIG. 23D
TO FIG. 23F
IDEENHI- 24 IDEENHI-
IDEENLO- 23 IDEENLO-
IOCS16- 27 IOCS16-
IDE I/F
HDCS1- 26 HDCS1-
HDCS0- 25 HDCS0-
IDED7 22 IDED7
ONLINE 59 SLCT
PE 60 PE
BUSY 61 BUSY
ACK- 62 ACK-
FAULT- 75 ERROR-
PARALLEL PORT
6 GND
47 GND
67 GND
95 GND
PD7 63 PD7
PD6 64 PD6
PD5 65 PD5
PD4 66 PD4
PD3 68 PD3
PD2 69 PD2
PD1 70 PD1
PD0 71 PD0
PD<0:7>
SLCTIN- 73 SELECT-
INIT- 74 INIT-
AUTOFD- 76 AUTOLF-
STROBE- 77 STROBE-
U15
80 51
81 50
U160 PINOUT
100 31
1 30

FIG. 23E



FROM FIG. 23C
VCC
VCC
C1503
10
10V
1206
42
41
44
20
21
29
34
28
IRQ5
IRQ5
[11]
TXD1-
TXD1-
[16]
RTS1-
RTS1-
[16]
DTR1-
DTR1-
[16]
ABT245PW
IOR-
SD6
SD5
SD4
SD3
SD2
SD1
SD0
GND
IDEENLO-
IDEENHI-
A1
A2
A3
A4
A5
A6
A7
A8
B1
B2
B3
B4
B5
B6
B7
B8
HDD6
HDD5
HDD4
HDD3
HDD2
HDD1
HDD0
U153
FROM FIG. 23E
HDCS1-
HDCS0-
IDED7
PD<0:7>
SELECT-
INIT-
AUTOLF-
STROBE-
VCC
C151
C152
C153
.1

FIG. 23F

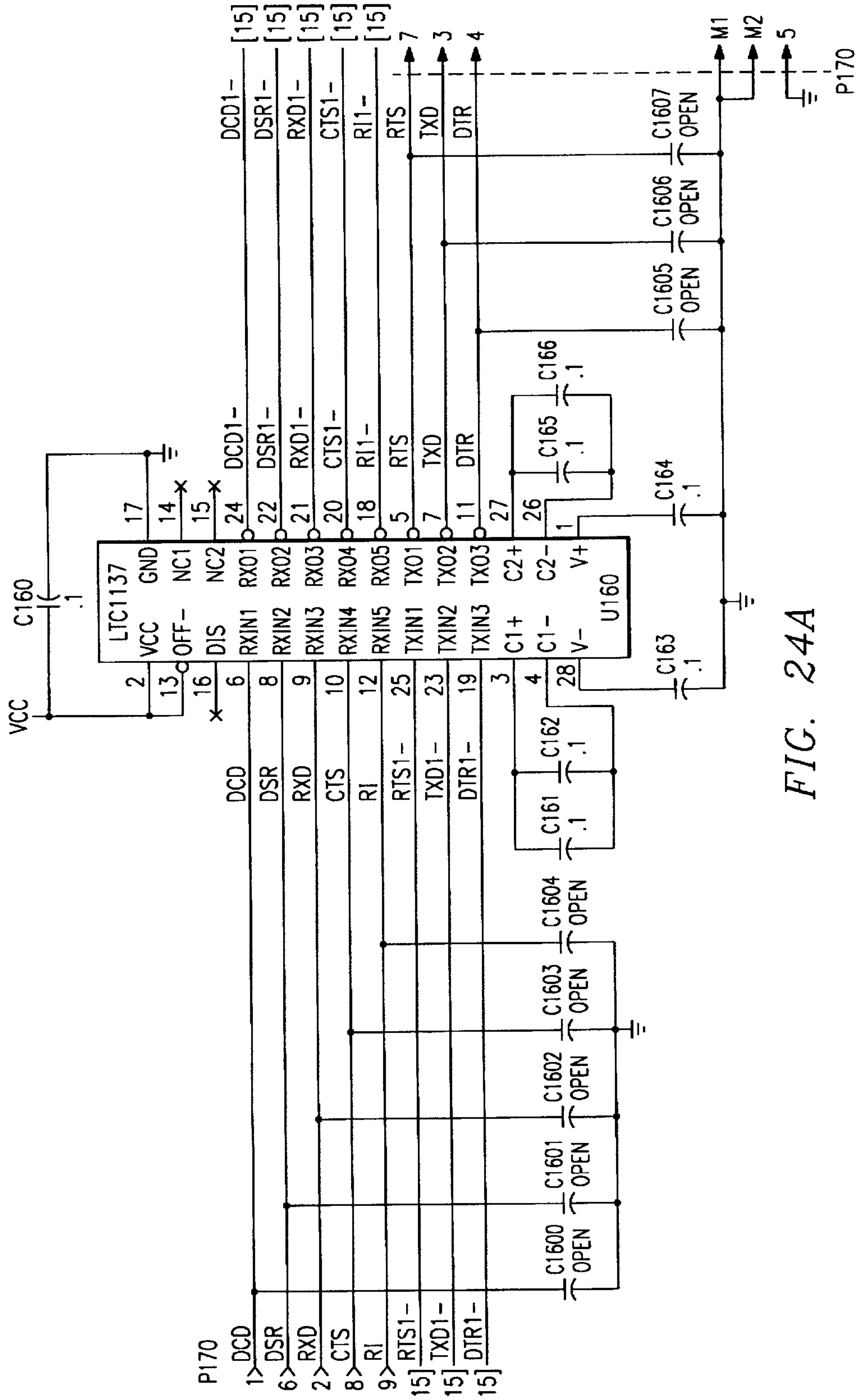

LTC1137
U160
VCC
OFF-
DIS
RXIN1
RXIN2
RXIN3
RXIN4
RXIN5
TXIN1
TXIN2
TXIN3
C1+
C1-
V-
GND
NC1
NC2
RXO1
RXO2
RXO3
RXO4
RXO5
TXO1
TXO2
TXO3
C2+
C2-
V+
C160
.1
C161
C162
C163
C164
C165
C166
C1600
C1601
C1602
C1603
C1604
C1605
C1606
C1607
OPEN
P170
DCD
DSR
RXD
CTS
RI
RTS1-
TXD1-
DTR1-
DCD1-
DSR1-
RXD1-
CTS1-
RI1-
RTS
TXD
DTR
[15]
M1
M2

FIG. 24A

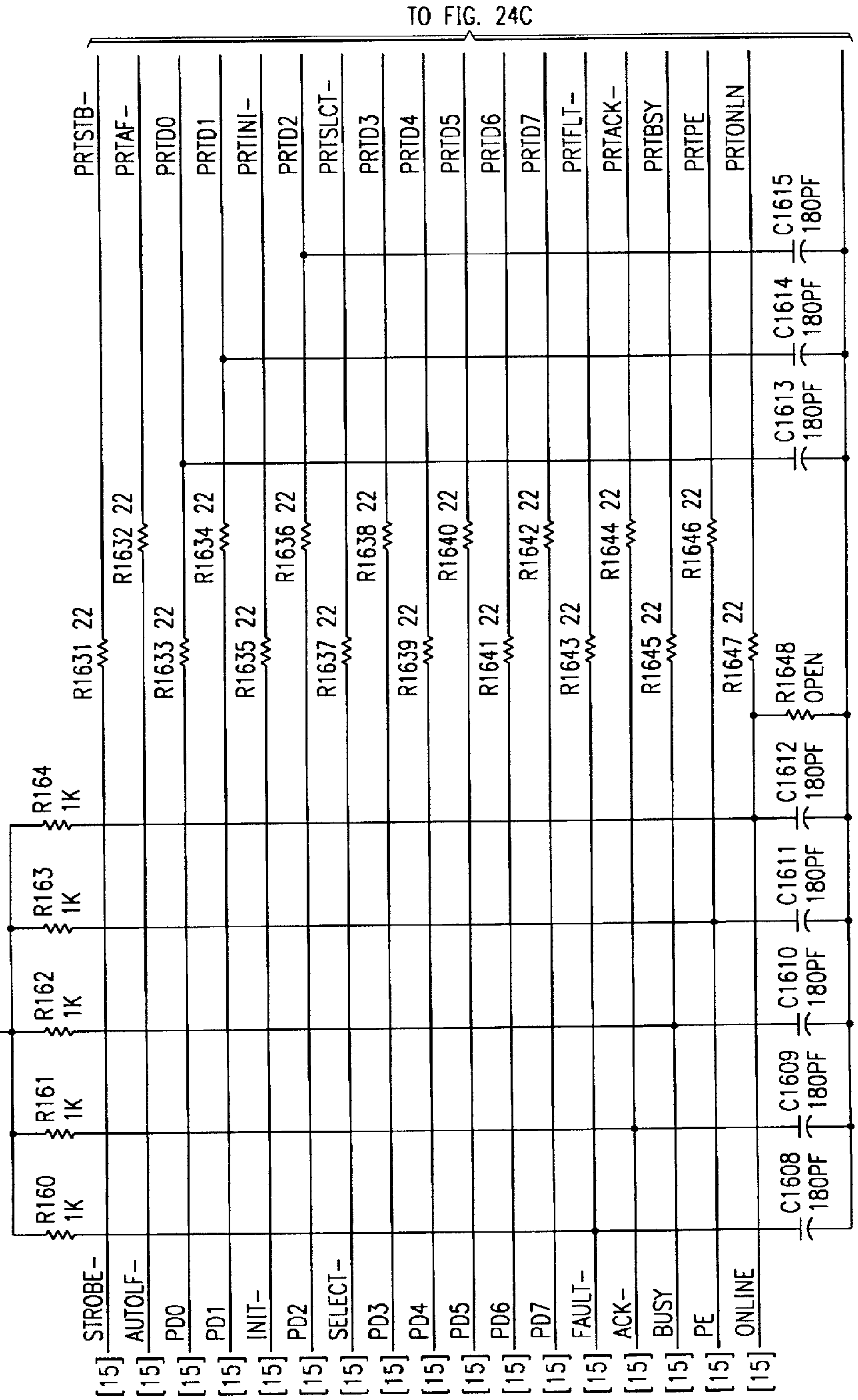

TO FIG. 24C
PRTSTB-
PRTAF-
PRTD0
PRTD1
PRTINI-
PRTD2
PRTSLCT-
PRTD3
PRTD4
PRTD5
PRTD6
PRTD7
PRTFLT-
PRTACK-
PRTBSY
PRTPE
PRTONLN
C1615 180PF
C1614 180PF
C1613 180PF
R1632 22
R1634 22
R1636 22
R1638 22
R1640 22
R1642 22
R1644 22
R1646 22
R1631 22
R1633 22
R1635 22
R1637 22
R1639 22
R1641 22
R1643 22
R1645 22
R1647 22
R1648 OPEN
R164 1K
R163 1K
R162 1K
R161 1K
R160 1K
VCC
C1612 180PF
C1611 180PF
C1610 180PF
C1609 180PF
C1608 180PF
STROBE-
AUTOLF-
PD0
PD1
INIT-
PD2
SELECT-
PD3
PD4
PD5
PD6
PD7
FAULT-
ACK-
BUSY
PE
ONLINE
[15]

FIG. 24B

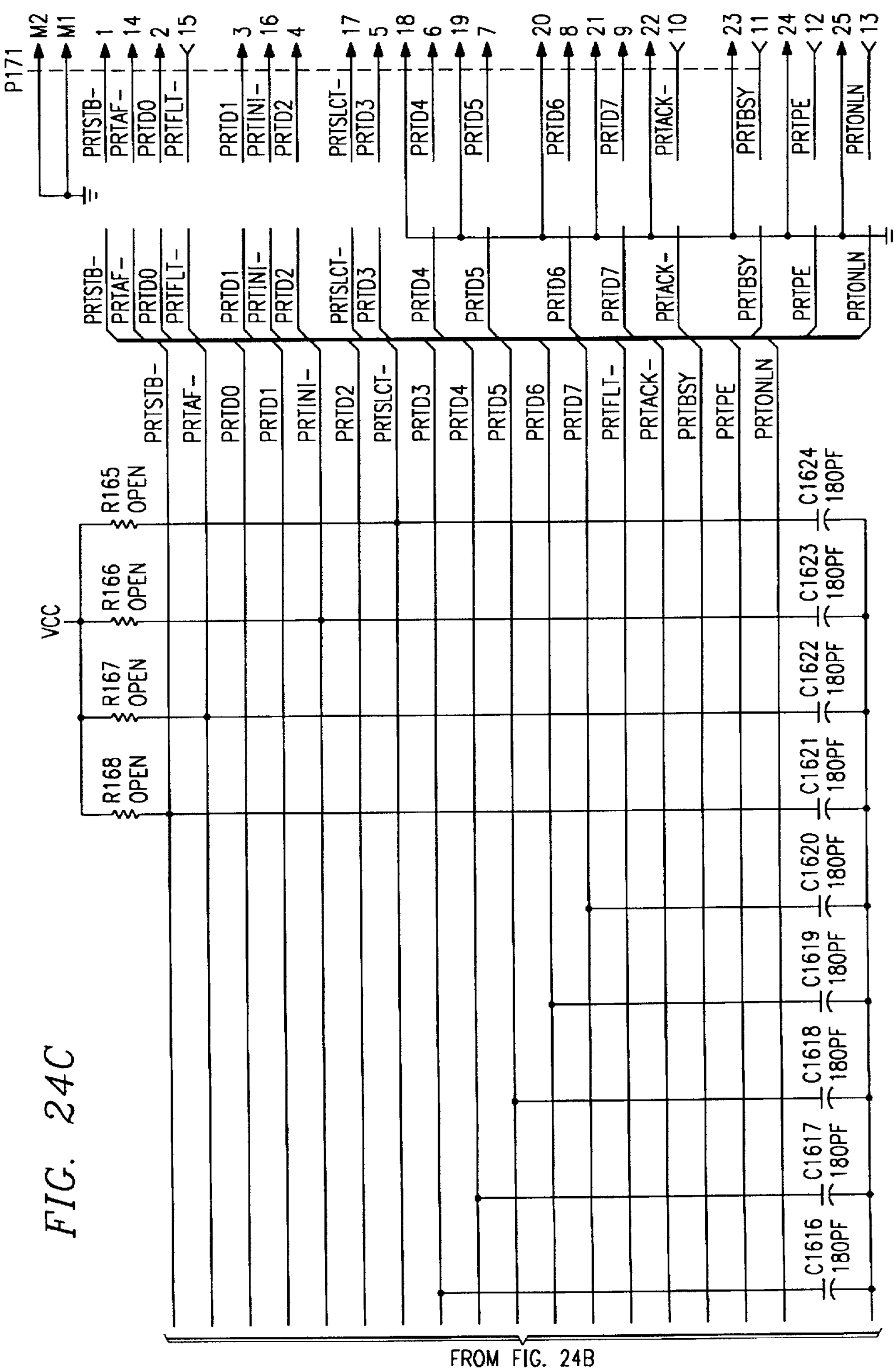
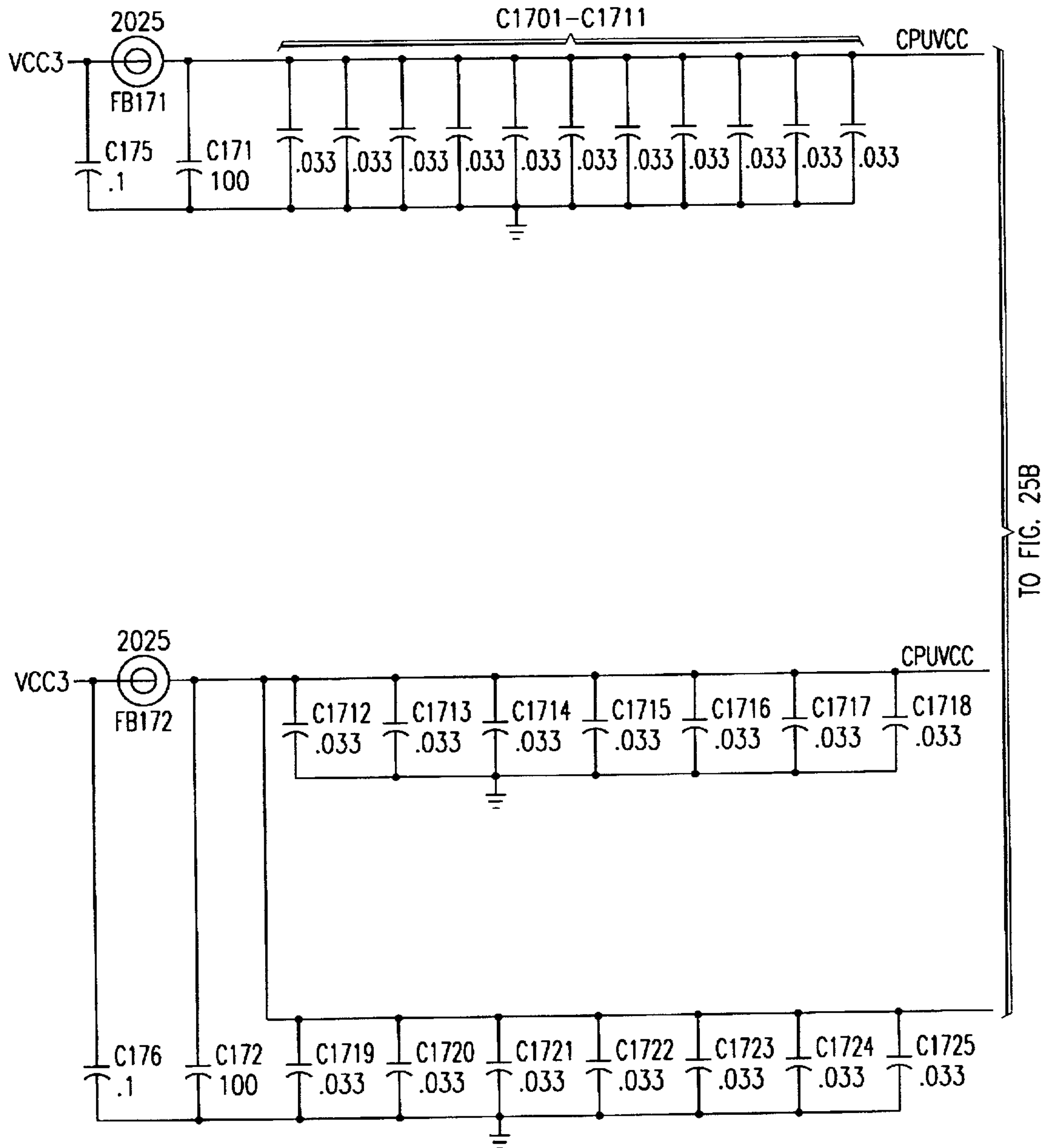
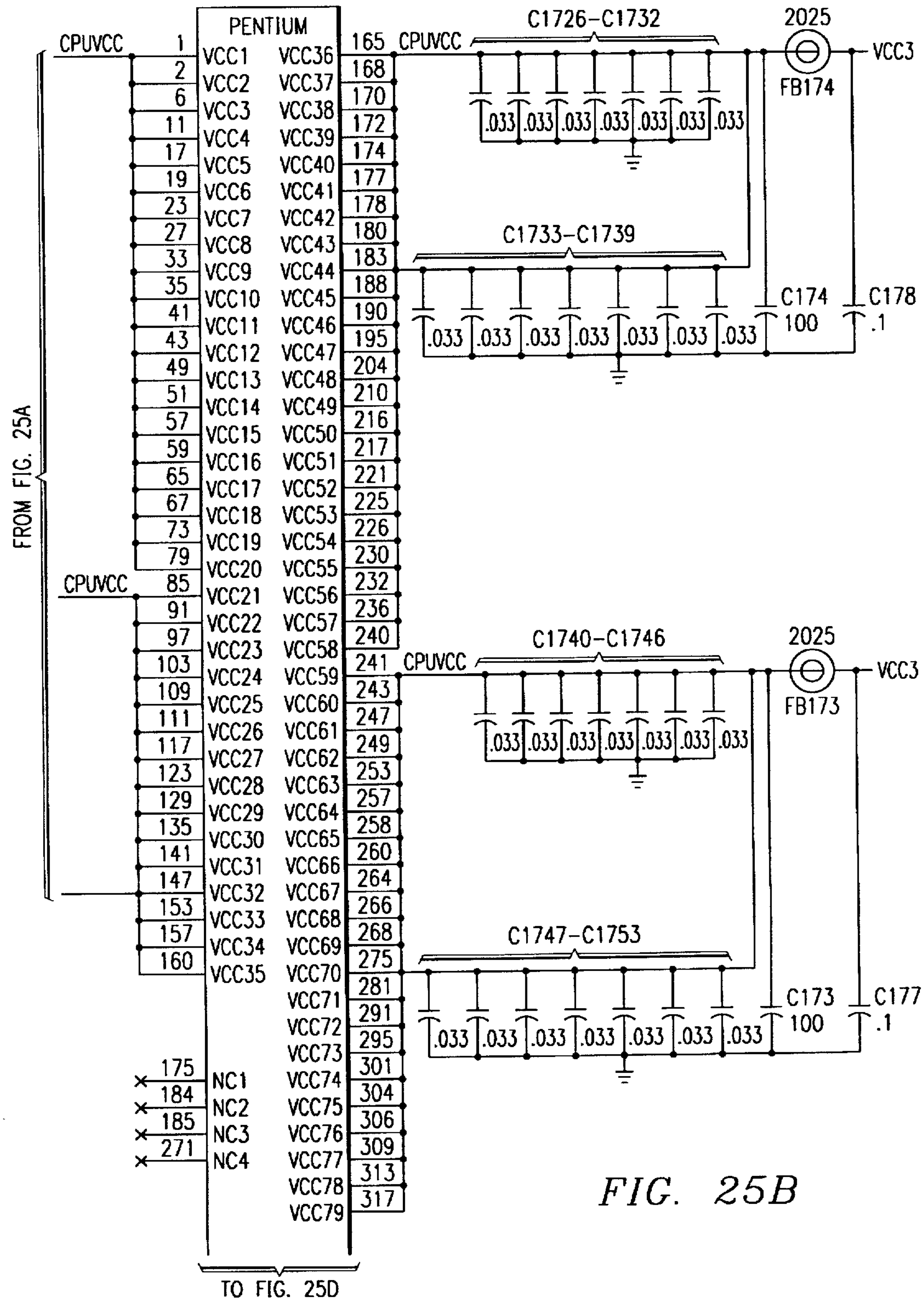

P171
M2
M1
PRTSTB-
PRTAF-
PRTD0
PRTFLT-
PRTD1
PRTINI-
PRTD2
PRTSLCT-
PRTD3
PRTD4
PRTD5
PRTD6
PRTD7
PRTACK-
PRTBSY
PRTPE
PRTONLN
VCC
R165 OPEN
R166 OPEN
R167 OPEN
R168 OPEN
C1616 180PF
C1617 180PF
C1618 180PF
C1619 180PF
C1620 180PF
C1621 180PF
C1622 180PF
C1623 180PF
C1624 180PF
FROM FIG. 24B

FIG. 24C



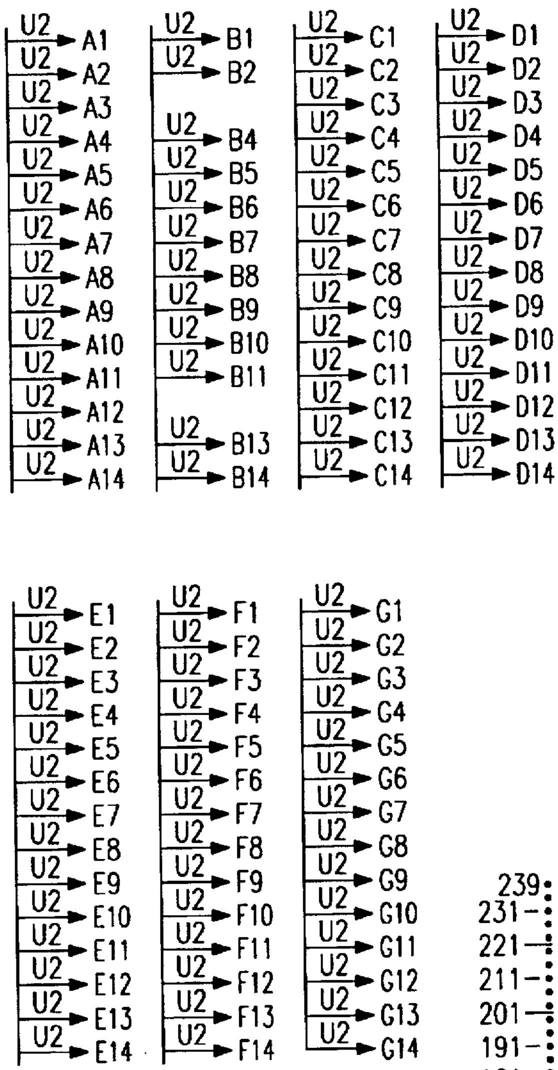
2025
VCC3
FB171
C1701-C1711
CPUVCC
C175
.1
C171
100
.033
.033
.033
.033
.033
.033
.033
.033
.033
.033
.033
TO FIG. 25B
2025
VCC3
FB172
CPUVCC
C1712
.033
C1713
.033
C1714
.033
C1715
.033
C1716
.033
C1717
.033
C1718
.033
C176
.1
C172
100
C1719
.033
C1720
.033
C1721
.033
C1722
.033
C1723
.033
C1724
.033
C1725
.033

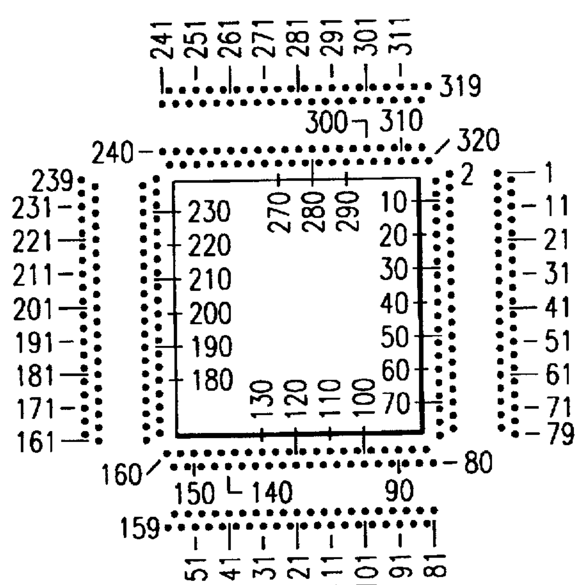
FIG. 25A



PENTIUM
CPUVCC
FROM FIG. 25A
TO FIG. 25D
C1726-C1732
C1733-C1739
C1740-C1746
C1747-C1753
2025
FB174
FB173
VCC3
C174
100
C178
.1
C173
C177
.033

FIG. 25B





PENTIUM PINOUT TCP 320
BACKSIDE VIEW (TOP OF PWB)

FIG. 25C

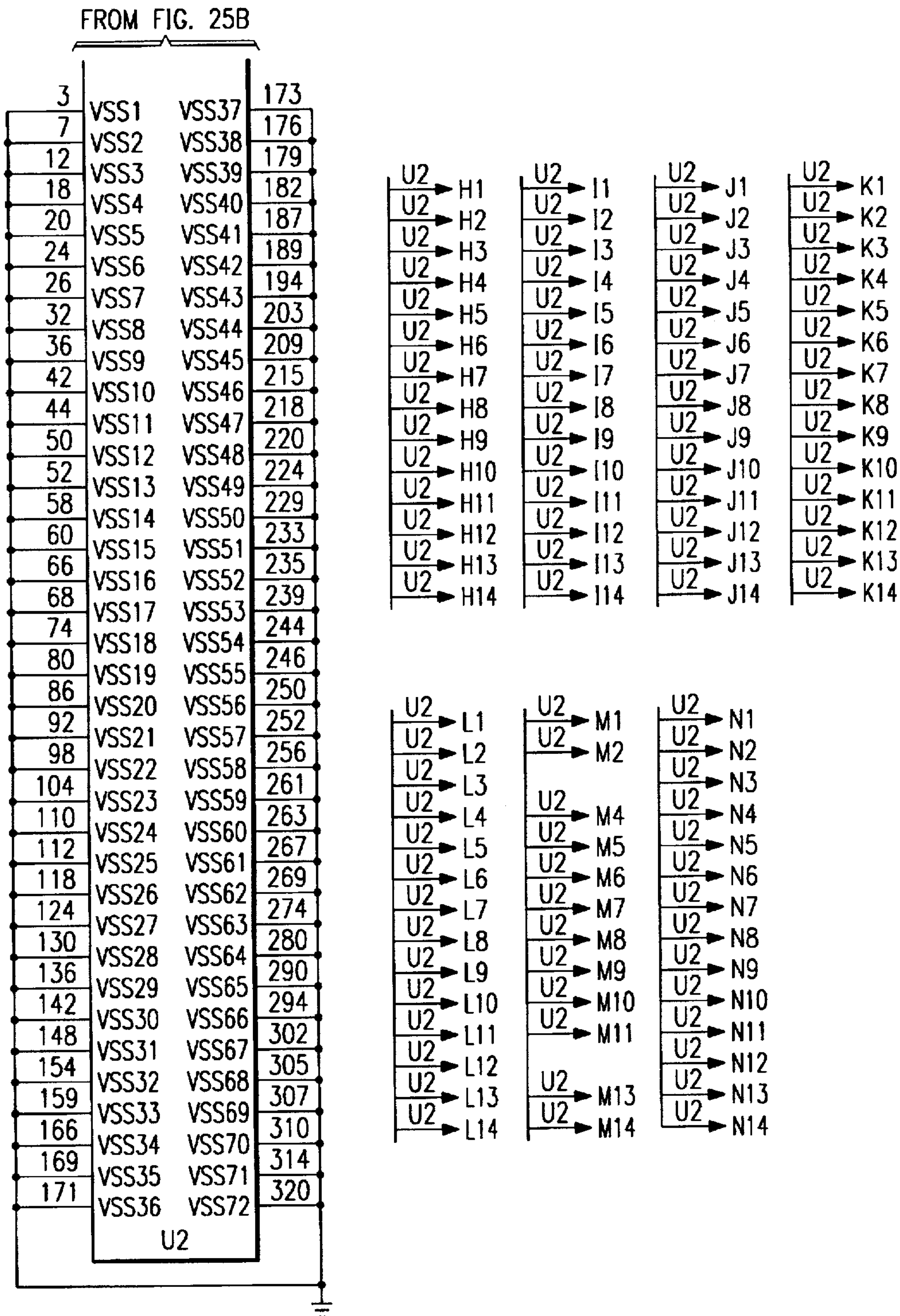
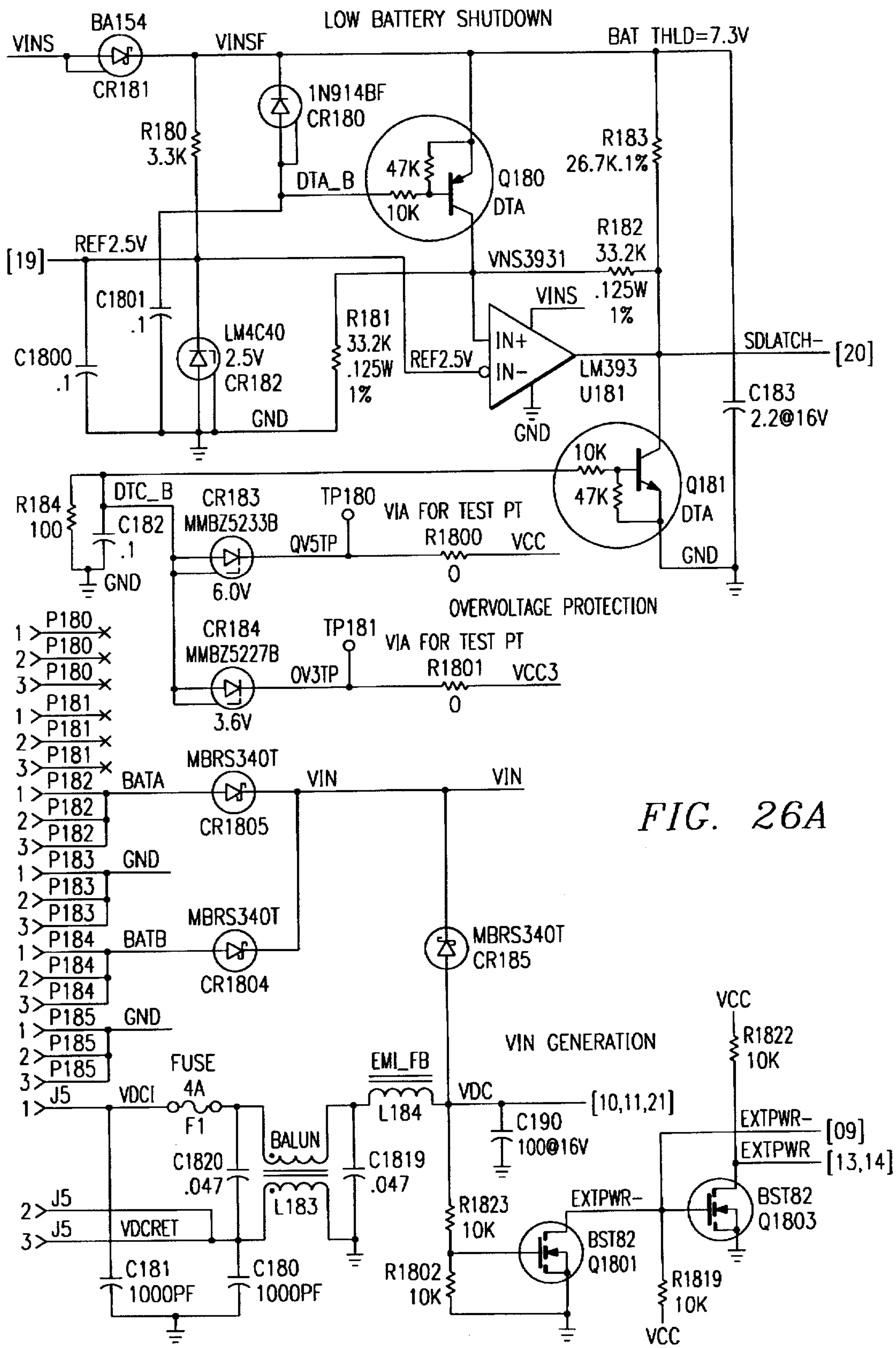

FROM FIG. 25B
U2
VSS1 VSS2 VSS3 VSS4 VSS5 VSS6 VSS7 VSS8 VSS9 VSS10 VSS11 VSS12 VSS13 VSS14 VSS15 VSS16 VSS17 VSS18 VSS19 VSS20 VSS21 VSS22 VSS23 VSS24 VSS25 VSS26 VSS27 VSS28 VSS29 VSS30 VSS31 VSS32 VSS33 VSS34 VSS35 VSS36
VSS37 VSS38 VSS39 VSS40 VSS41 VSS42 VSS43 VSS44 VSS45 VSS46 VSS47 VSS48 VSS49 VSS50 VSS51 VSS52 VSS53 VSS54 VSS55 VSS56 VSS57 VSS58 VSS59 VSS60 VSS61 VSS62 VSS63 VSS64 VSS65 VSS66 VSS67 VSS68 VSS69 VSS70 VSS71 VSS72
3 7 12 18 20 24 26 32 36 42 44 50 52 58 60 66 68 74 80 86 92 98 104 110 112 118 124 130 136 142 148 154 159 166 169 171
173 176 179 182 187 189 194 203 209 215 218 220 224 229 233 235 239 244 246 250 252 256 261 263 267 269 274 280 290 294 302 305 307 310 314 320
U2 H1 H2 H3 H4 H5 H6 H7 H8 H9 H10 H11 H12 H13 H14
U2 I1 I2 I3 I4 I5 I6 I7 I8 I9 I10 I11 I12 I13 I14
U2 J1 J2 J3 J4 J5 J6 J7 J8 J9 J10 J11 J12 J13 J14
U2 K1 K2 K3 K4 K5 K6 K7 K8 K9 K10 K11 K12 K13 K14
U2 L1 L2 L3 L4 L5 L6 L7 L8 L9 L10 L11 L12 L13 L14
U2 M1 M2 M4 M5 M6 M7 M8 M9 M10 M11 M13 M14
U2 N1 N2 N3 N4 N5 N6 N7 N8 N9 N10 N11 N12 N13 N14

FIG. 25D



LOW BATTERY SHUTDOWN
VINS
BA154
VINSF
CR181
1N914BF
CR180
R180
3.3K
BAT THLD=7.3V
R183
26.7K.1%
47K
10K
DTA_B
Q180
DTA
R182
33.2K
.125W
1%
REF2.5V
[19]
VNS3931
C1801
.1
C1800
.1
LM4C40
2.5V
CR182
R181
33.2K
.125W
1%
REF2.5V
IN+
IN-
LM393
U181
SDLATCH-
[20]
C183
2.2@16V
GND
R184
100
DTC_B
C182
.1
CR183
MMBZ5233B
6.0V
TP180
VIA FOR TEST PT
QV5TP
R1800
0
VCC
Q181
OVERVOLTAGE PROTECTION
CR184
MMBZ5227B
3.6V
TP181
QV3TP
R1801
VCC3
P180
P181
P182
P183
P184
P185
BATA
MBRS340T
CR1805
VIN
BATB
CR1804
MBRS340T
CR185
VIN GENERATION
VCC
R1822
10K
J5
VDCI
FUSE
4A
F1
EMI_FB
L184
VDC
[10,11,21]
C190
100@16V
EXTPWR-
[09]
EXTPWR
[13,14]
BALUN
C1820
.047
L183
C1819
.047
R1823
10K
VDCRET
C181
1000PF
C180
1000PF
R1802
10K
BST82
Q1801
BST82
Q1803
R1819
10K

FIG. 26A

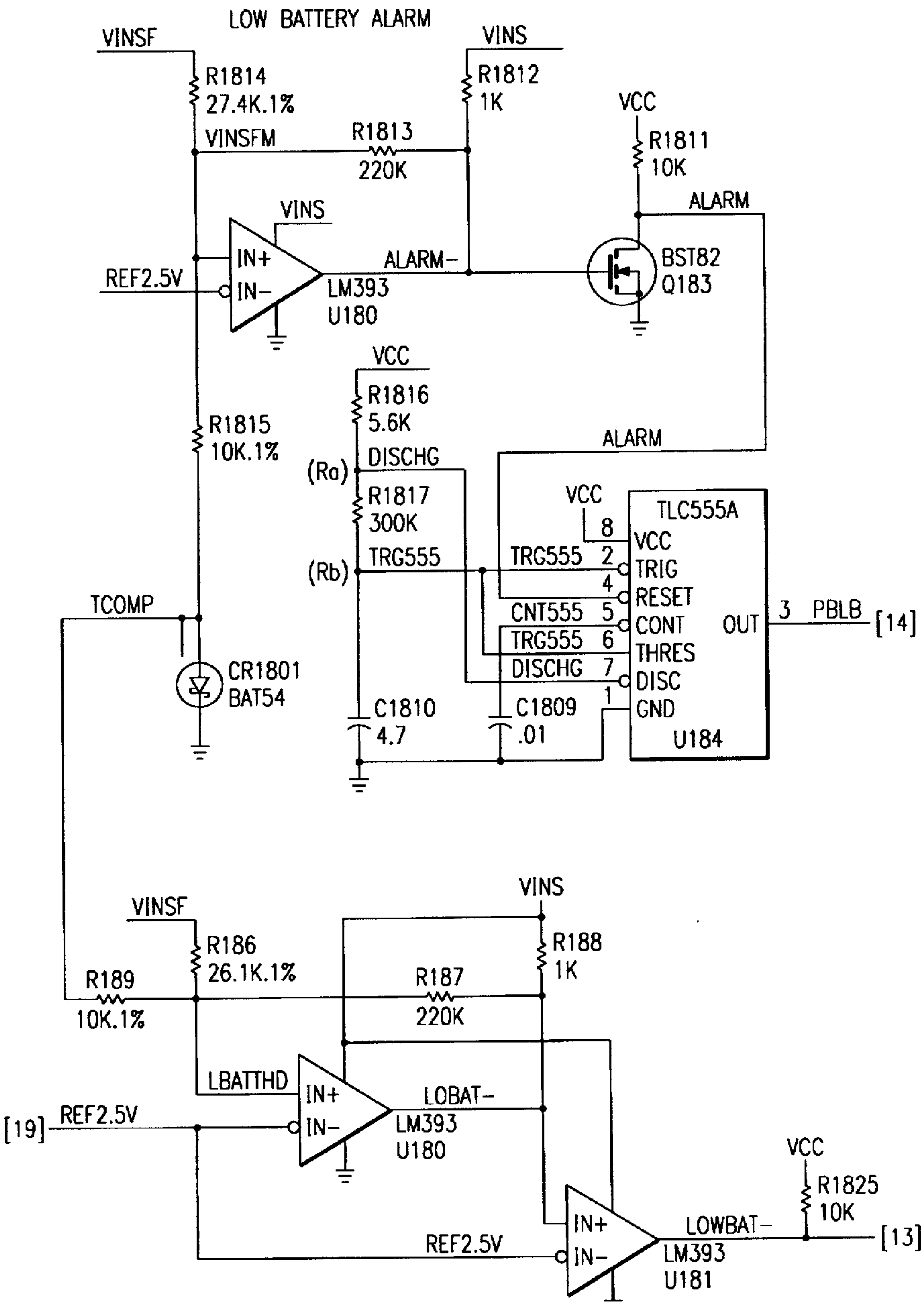

LOW BATTERY ALARM
VINSF
R1814
27.4K.1%
VINSFM
R1813
220K
VINS
R1812
1K
VCC
R1811
10K
ALARM
VINS
IN+
IN-
REF2.5V
ALARM-
LM393
U180
BST82
Q183
VCC
R1816
5.6K
DISCHG
(Ra)
R1817
300K
TRG555
(Rb)
R1815
10K.1%
ALARM
VCC
8
TLC555A
VCC
TRG555 2
TRIG
4
RESET
CNT555 5
CONT
OUT
3 PBLB
[14]
TRG555 6
THRES
DISCHG 7
DISC
1
GND
U184
TCOMP
CR1801
BAT54
C1810
4.7
C1809
.01
VINSF
R186
26.1K.1%
VINS
R188
1K
R189
10K.1%
R187
220K
LBATTHD
IN+
IN-
LOBAT-
LM393
U180
[19]
REF2.5V
VCC
R1825
10K
IN+
IN-
REF2.5V
LOWBAT-
LM393
U181
[13]

FIG. 26B

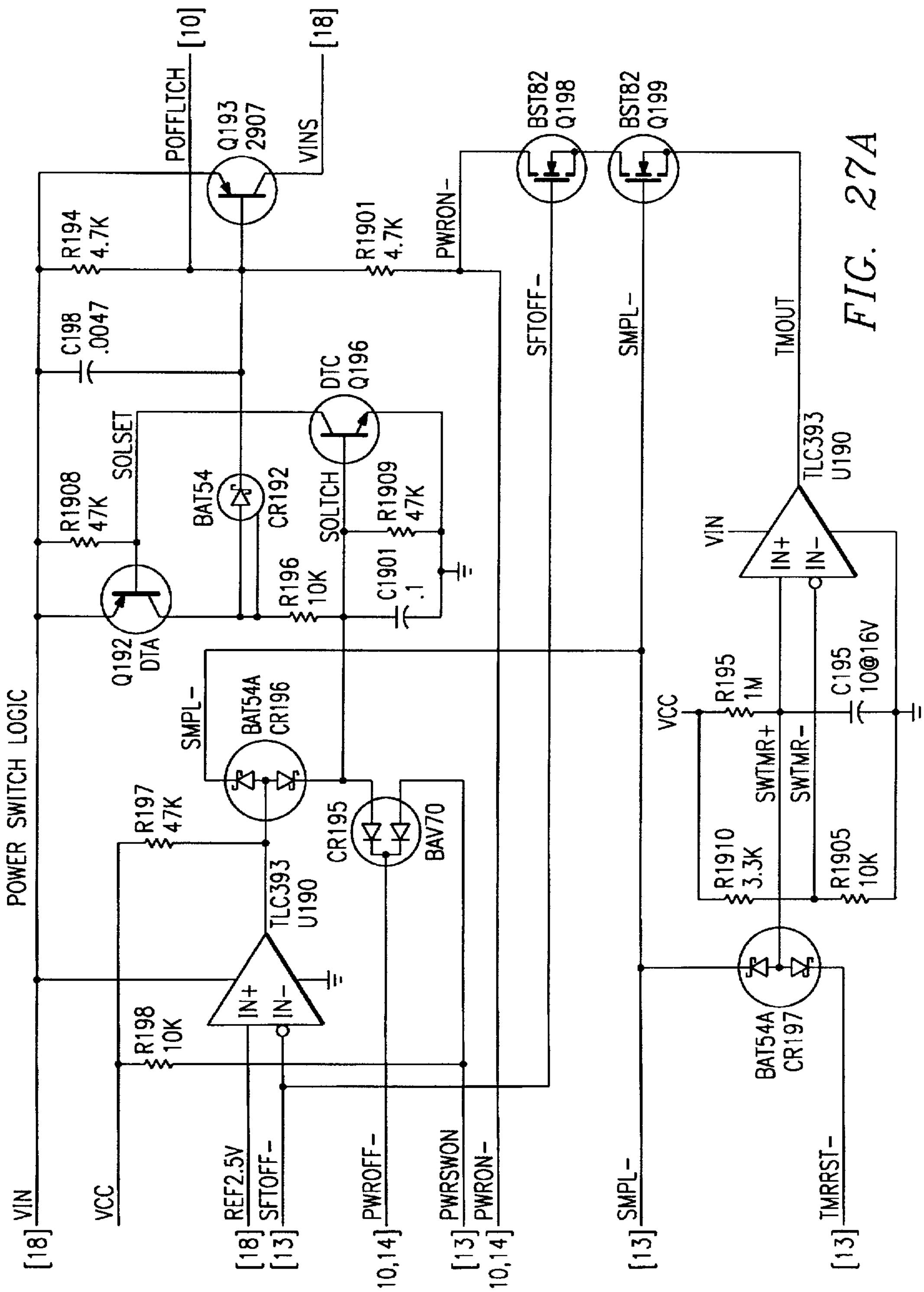
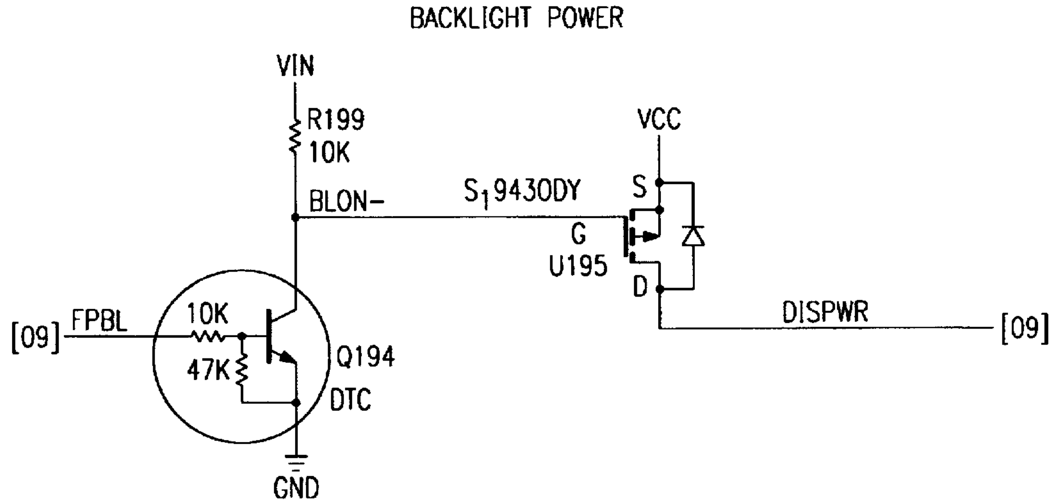

POWER SWITCH LOGIC
VIN
VCC
REF2.5V
SFTOFF-
PWROFF-
PWRSWON
PWRON-
SMPL-
TMRRST-
[18]
[13]
[10,14]
[10]
POFFLTCH
Q193
2907
VINS
R194
4.7K
R1901
4.7K
C198
.0047
R1908
47K
SOLSET
Q192
DTA
BAT54
CR192
R196
10K
DTC
Q196
SOLTCH
R1909
47K
C1901
.1
BAT54A
CR196
R197
47K
TLC393
U190
IN+
IN-
R198
10K
CR195
BAV70
BST82
Q198
Q199
SFTOFF-
TMOUT
R195
1M
C195
10@16V
SWTMR+
SWTMR-
R1910
3.3K
R1905
10K
BAT54A
CR197

FIG. 27A


BACKLIGHT POWER
VIN
R199
10K
VCC
BLON-
S1 9430DY
S
G
U195
D
DISPWR
[09]
[09]
FPBL
10K
47K
Q194
DTC
GND

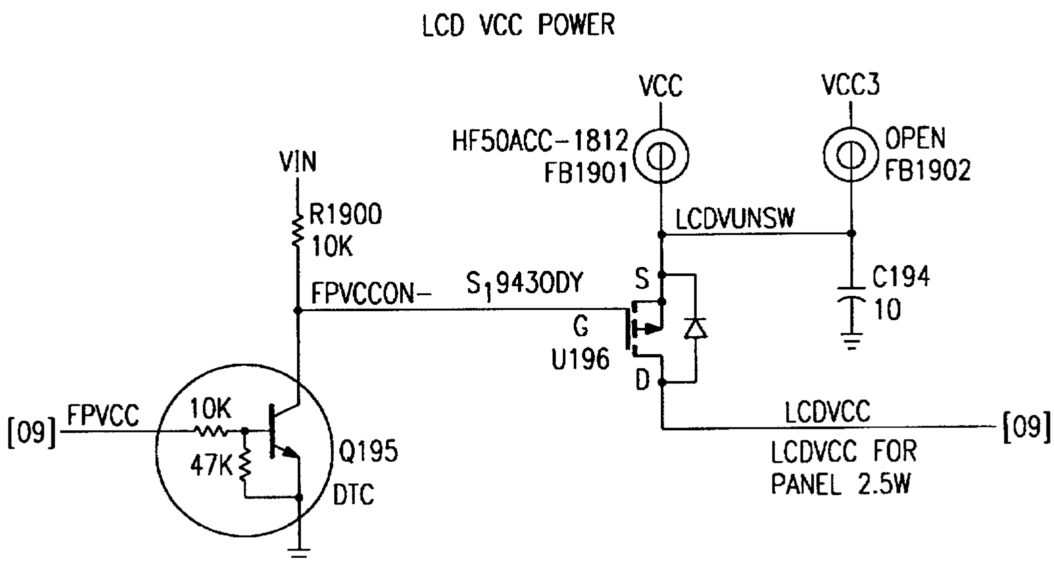
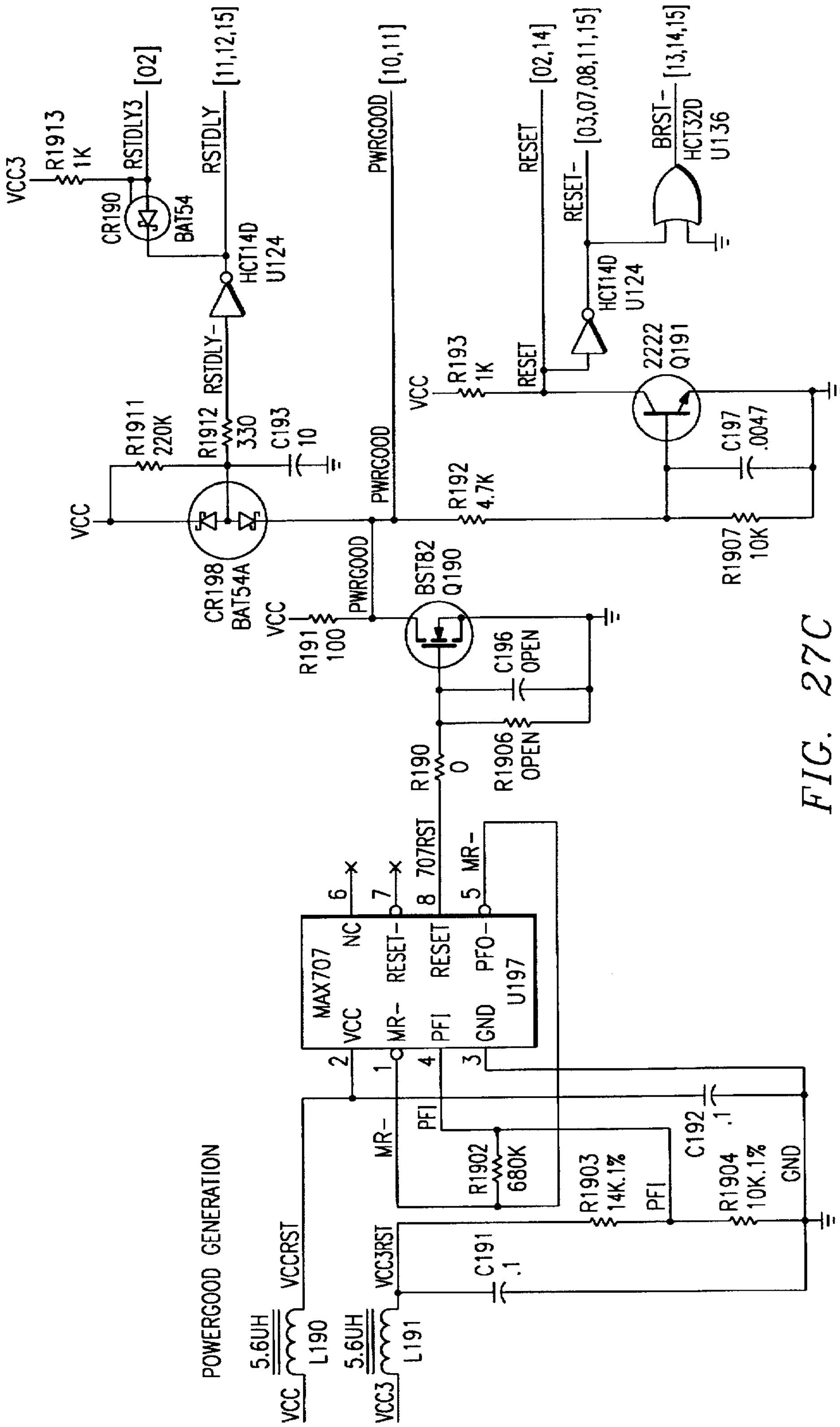
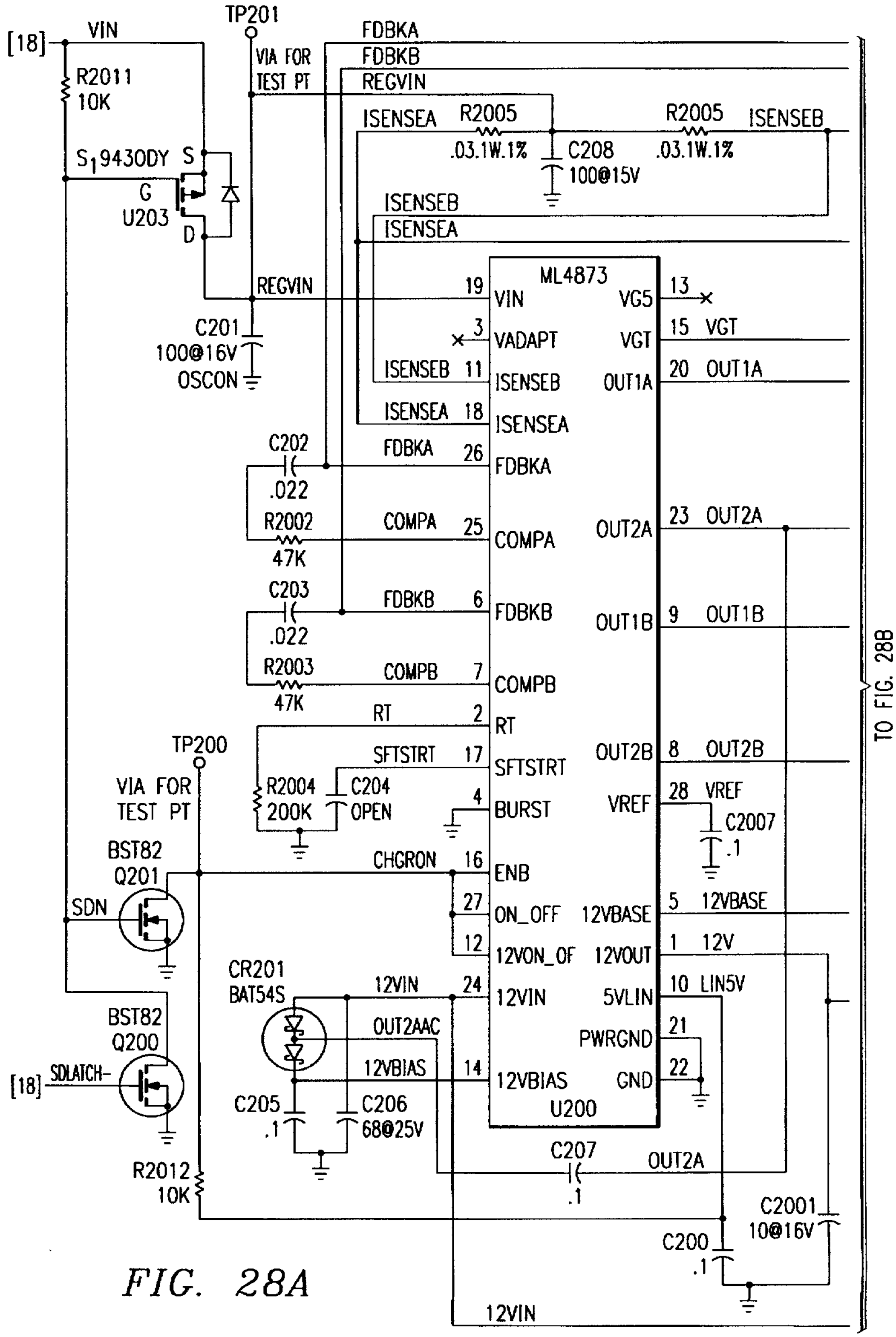

LCD VCC POWER
VCC
VCC3
HF50ACC-1812
FB1901
OPEN
FB1902
VIN
R1900
10K
LCDVUNSW
C194
10
FPVCCON-
S1 9430DY
S
G
U196
D
LCDVCC
[09]
[09]
FPVCC
10K
47K
Q195
DTC
LCDVCC FOR
PANEL 2.5W

FIG. 27B



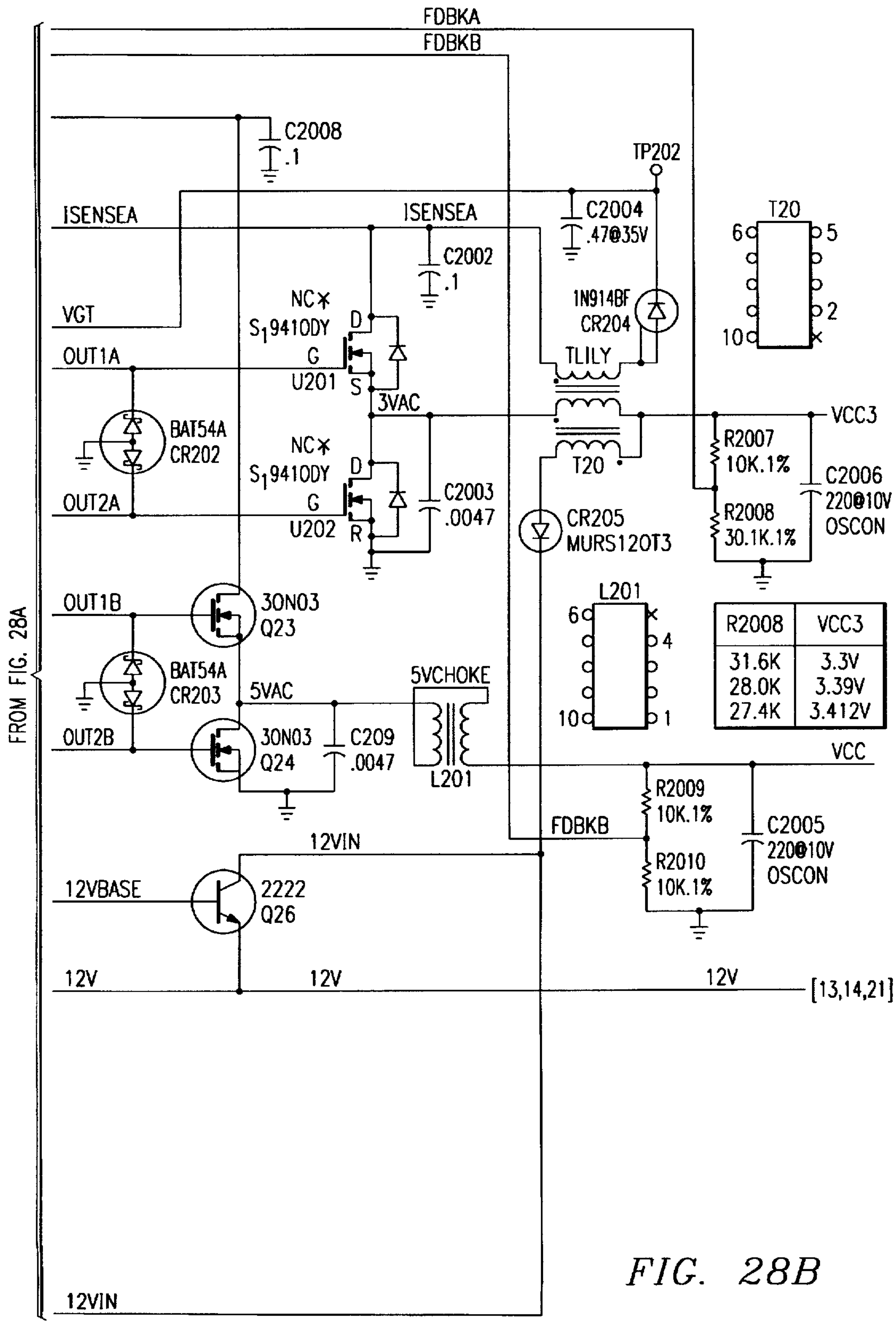
VCC3
R1913
1K
RSTDLY3
[02]
CR190
BAT54
RSTDLY
[11,12,15]
HCT14D
U124
RSTDLY-
R1911
220K
R1912
330
C193
10
VCC
CR198
BAT54A
PWRGOOD
[10,11]
VCC
R193
1K
RESET
[02,14]
RESET-
[03,07,08,11,15]
BRST-
[13,14,15]
HCT32D
U136
HCT14D
U124
2222
Q191
C197
.0047
R192
4.7K
R1907
10K
VCC
R191
100
BST82
Q190
C196
OPEN
R190
0
R1906
OPEN
707RST
MR-
MAX707
VCC
MR-
PFI
GND
NC
RESET-
RESET
PFO-
U197
PFI
R1902
680K
R1903
14K.1%
R1904
10K.1%
C192
.1
GND
POWERGOOD GENERATION
VCCRST
5.6UH
L190
VCC3RST
C191
.1
5.6UH
L191
VCC3

FIG. 27C



[18]
VIN
TP201
VIA FOR TEST PT
FDBKA
FDBKB
REGVIN
R2011 10K
ISENSEA
R2005 .03.1W.1%
C208 100@15V
R2005 .03.1W.1%
ISENSEB
S1 9430DY
S
G
U203
D
ISENSEB
ISENSEA
ML4873
REGVIN
19 VIN
VG5 13
3 VADAPT
VGT 15 VGT
C201 100@16V OSCON
ISENSEB 11 ISENSEB
OUT1A 20 OUT1A
ISENSEA 18 ISENSEA
C202 .022
FDBKA 26 FDBKA
R2002 47K
COMPA 25 COMPA
OUT2A 23 OUT2A
C203 .022
FDBKB 6 FDBKB
OUT1B 9 OUT1B
R2003 47K
COMPB 7 COMPB
RT 2 RT
TP200
VIA FOR TEST PT
SFTSTRT 17 SFTSTRT
OUT2B 8 OUT2B
R2004 200K
C204 OPEN
4 BURST
VREF 28 VREF
C2007 .1
BST82 Q201
CHGRON 16 ENB
SDN
27 ON_OFF
12VBASE 5 12VBASE
12 12VON_OF
12VOUT 1 12V
CR201 BAT54S
12VIN 24 12VIN
5VLIN 10 LIN5V
BST82 Q200
OUT2AAC
PWRGND 21
[18] SDLATCH-
12VBIAS 14 12VBIAS
GND 22
U200
C205 .1
C206 68@25V
C207 .1
OUT2A
R2012 10K
C2001 10@16V
C200 .1
12VIN
TO FIG. 28B

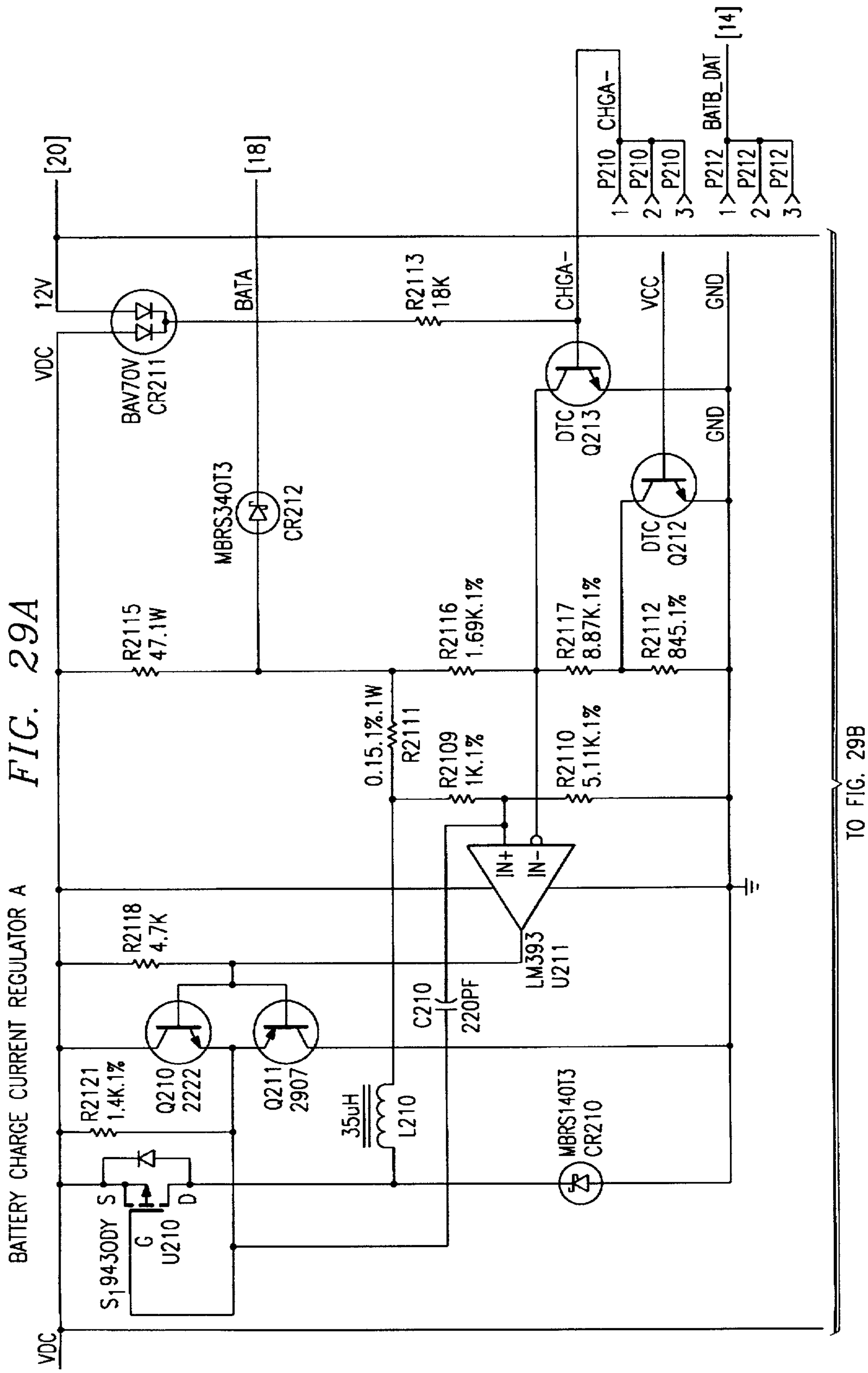
FIG. 28A



R2008	VCC3
31.6K	3.3V
28.0K	3.39V
27.4K	3.412V

FIG. 28B

FIG. 29A


BATTERY CHARGE CURRENT REGULATOR A
[20]
12V
VDC
BAV70V
CR211
BATA
[18]
MBRS340T3
CR212
R2115
47.1W
R2113
18K
CHGA-
DTC
Q213
VCC
GND
DTC
Q212
R2116
1.69K.1%
R2117
8.87K.1%
R2112
845.1%
0.15.1%.1W
R2111
R2109
1K.1%
R2110
5.11K.1%
IN+
IN-
LM393
U211
R2118
4.7K
C210
220PF
Q210
2222
Q211
2907
R2121
1.4K.1%
35uH
L210
MBRS140T3
CR210
S1943ODY
S
G
D
U210
P210
P212
CHGA-
BATB_DAT
[14]
TO FIG. 29B

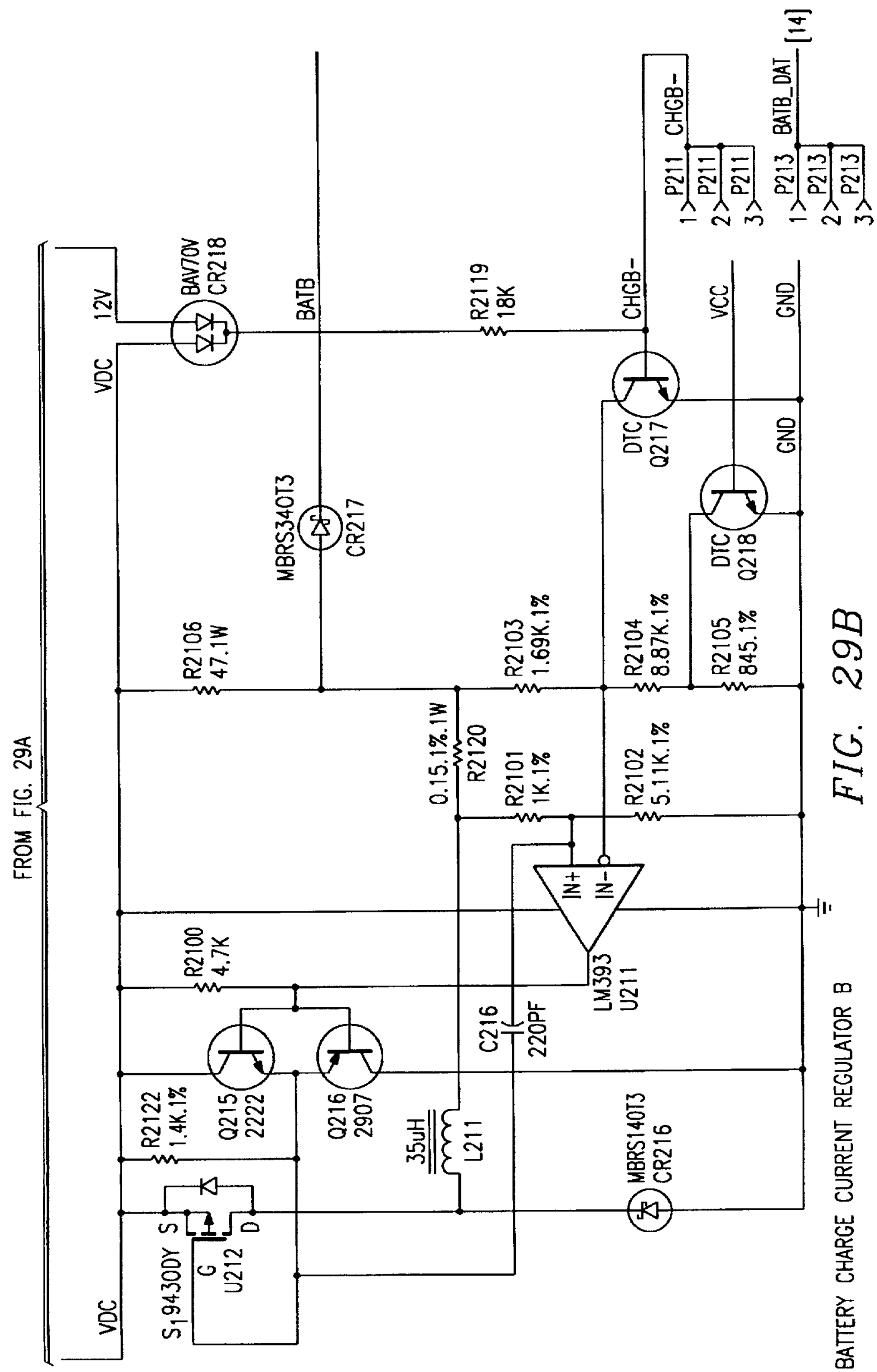
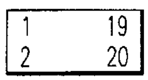

FROM FIG. 29A
VDC
12V
VDC
BAV70V
CR218
BATB
R2119
18K
CHGB-
P211 CHGB-
P211
P211
P213 BATB_DAT
P213
P213
[14]
VCC
GND
GND
DTC
Q217
DTC
Q218
MBRS340T3
CR217
R2106
47.1W
R2103
1.69K.1%
R2104
8.87K.1%
R2105
845.1%
0.15.1%.1W
R2120
R2101
1K.1%
R2102
5.11K.1%
IN+
IN-
LM393
U211
R2100
4.7K
C216
220PF
Q215
2222
Q216
2907
R2122
1.4K.1%
35uH
L211
MBRS140T3
CR216
S
D
G
U212
S1 9430DY
BATTERY CHARGE CURRENT REGULATOR B

FIG. 29B

P91 LCD

PIN#	SIGNAL	SH#	PIN#	SIGNAL	SH#
1	GND	09	11	FP7	09
2	FPVDCLK (SCLK)	09	12	VCC	09
3	GND	09	13	FP6	09
4	EXTPWR-	09	14	GND	09
5	FP12	09	15	FP5	09
6	FP2	09	16	FP4	09
7	FP11	09	17	FPDE	09
8	GND	09	18	FP3	09
9	FP10	09	19	LFS (FLM)	09
10	FPVEE	09	20	LLCLK (LP)	09

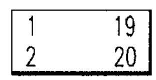


20 PIN DOUBLE ROW HEADER

SIGNAL	9BIT TFT	12BIT TFT	18BIT TFT	16BIT STN
FP2	N.C.	N.C.	B0	N.C.
FP3	N.C.	N.C.	B1	MOD
FP4	N.C.	B0	B2	LD0
FP5	B0	B1	B3	LD1
FP6	B1	B2	B4	LD2
FP7	B2	B3	B5	LD3
FP10	N.C.	N.C.	G0	UD4
FP11	N.C.	N.C.	G1	UD5
FP12	N.C.	G0	G2	LD4

P92 LCD

PIN#	SIGNAL	SH#	PIN#	SIGNAL	SH#
1	FP23	09	11	FP20	09
2	DISPWR	09	12	PNLTYP0	09
3	GND	09	13	FP19	09
4	DISPWR	09	14	PNLTYP1	09
5	FP22	09	15	FP18	09
6	LCDVCC	09	16	PNLTYP2	09
7	GND	09	17	FP15	09
8	LCDVCC	09	18	GND	09
9	FP21	09	19	FP14	09
10	GND	09	20	FP13	09



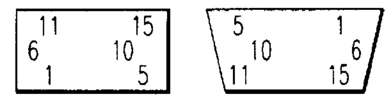
20 PIN DOUBLE ROW HEADER

SIGNAL	9BIT TFT	12BIT TFT	18BIT TFT	16BIT STN
FP13	G0	G1	G3	LD5
FP14	G1	G2	G4	LD6
FP15	G2	G3	G5	LD7
FP18	N.C.	N.C.	R0	UD6
FP19	N.C.	N.C.	R1	UD7
FP20	N.C.	R0	R2	UD0
FP21	R0	R1	R3	UD1
FP22	R1	R2	R4	UD2
FP23	R2	R3	R5	UD3

P90

PIN#	SIGNAL	SH#	PIN#	SIGNAL	SH#
1	CBLUE	09	9	N.C.	09
2	CGREEN	09	10	GND	09
3	CRED	09	11	N.C.	09
4	N.C.	09	12	N.C.	09
5	GND	09	13	CHSYNC	09
6	AGND	09	14	CVSYNC	09
7	AGND	09	15	N.C.	09
8	AGND	09			

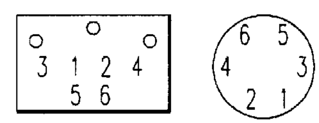
PWB TOP SODE INTO FACE OF CONN



15 PIN D SUB RECEPTACLE

P142 MOUSE

PIN#	SIGNAL	SH#
1	PSDATA	14
2	N.C.	14
3	PSGND	14
4	PSVCC	14
5	PSCLK	14
6	N.C.	14



PWB TOP SIDE FACE OF CONNECTOR

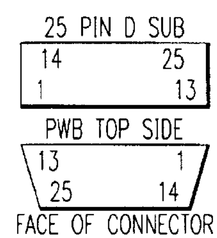
FIG. 30A

P40 MEMORY EXPANSION

PIN#	SIGNAL	SH#	PIN#	SIGNAL	SH#	PIN#	SIGNAL	SH#	PIN#	SIGNAL	SH#	PIN#	SIGNAL	SH#	PIN#	SIGNAL	SH#	PIN#	SIGNAL	SH#	PIN#	SIGNAL	SH#
L1	GND	04	L18	MD21	04	L35	VCC3	06	L52	GND	06	R1	GND	06	R18	MD63	06	R35	VCC3	06	R52	GND	06
L2	MD1	04	L19	MD23	04	L36	MD0	06	L53	MD22	06	R2	MD43	06	R19	N.C.	06	R36	MD42	06	R53	N.C.	06
L3	GND	04	L20	MD24	04	L37	MD2	06	L54	GND	06	R3	GND	06	R20	GND	06	R37	MD44	06	R54	VCC3	06
L4	MD4	04	L21	MD26	04	L38	MD3	06	L55	MD25	06	R4	MD46	06	R21	RAS2-	06	R38	MD45	06	R55	RAS3-	06
L5	GND	04	L22	MD27	04	L39	MD5	06	L56	GND	06	R5	GND	06	R22	RAS1-	06	R39	MD47	06	R56	GND	06
L6	CAS0-	04	L23	MD29	04	L40	CAS1-	06	L57	MD28	06	R6	CAS4-	06	R23	GND	06	R40	MD57	06	R57	RAS0-	06
L7	GND	04	L24	CAS3-	04	L41	WEN-	06	L58	GND	06	R7	MD48	06	R24	CAS6-	06	R41	GND	06	R58	CAS7-	06
L8	MD7	04	L25	GND	04	L42	MD6	06	L59	CAS2-	06	R8	MD50	06	R25	MA4	06	R42	MD49	06	R59	GND	06
L9	VCC3	04	L26	MD30	04	L43	MD8	06	L60	VCC3	06	R9	VCC3	06	R26	GND	06	R43	GND	06	R60	MA5	06
L10	MD10	04	L27	MD32	04	L44	MD9	06	L61	MD31	06	R10	MD52	06	R27	MA7	06	R44	MD51	06	R61	MA6	06
L11	GND	04	L28	MD33	04	L45	MD11	06	L62	GND	06	R11	GND	06	R28	MA8	06	R45	MD53	06	R62	GND	06
L12	MD13	04	L29	MD35	04	L46	MD12	06	L63	MD34	06	R12	MD55	06	R29	GND	06	R46	MD54	06	R63	MA9	06
L13	GND	04	L30	MD36	04	L47	MD14	06	L64	GND	06	R13	GND	06	R30	MA11	06	R47	MD56	06	R64	MA10	06
L14	MD16	04	L31	MD38	04	L48	MD15	06	L65	MD37	06	R14	MD58	06	R31	MA3	06	R48	MD57	06	R65	GND	06
L15	VCC3	04	L32	MD39	04	L49	MD17	06	L66	GND	06	R15	VCC3	06	R32	GND	06	R49	MD59	06	R66	AM2	06
L16	MD18	04	L33	MD41	04	L50	GND	06	L67	MD40	06	R16	MD60	06	R33	MA0	06	R50	GND	06	R67	MA1	06
L17	MD20	04	L34	GND	04	L51	MD19	06	L68	VCC3	06	R17	MD62	06	R34	GND	06	R51	MD61	06	R68	VCC3	06

P171 PRINTER

PIN#	SIGNAL	SH#	PIN#	SIGNAL	SH#	PIN#	SIGNAL	SH#
1	PRTSTB-	16	10	PRTACK-	16	19	GND	16
2	PRTD0	16	11	PRTBSY	16	20	GND	16
3	PRTD1	16	12	PRTPE	16	21	GND	16
4	PRTD2	16	13	PRTONLN	16	22	GND	16
5	PRTD3	16	14	PRTAF-	16	23	GND	16
6	PRTD4	16	15	PRTFLT-	16	24	GND	16
7	PRTD5	16	16	PRTINI-	16	25	GND	16
8	PRTD6	16	17	PRTSLCT-	16			
9	PRTD7	16	18	GND	16			



P170 RS232

PIN#	SIGNAL	SH#
1	DCD	16
2	RXD	16
3	TXD	16
4	DTR	16
5	GND	16
6	DSR	16
7	RTS	16
8	CTS	16
9	RI	16

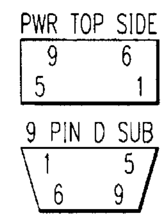


FIG. 30B

P140 PCMCIA/SOUND CARD

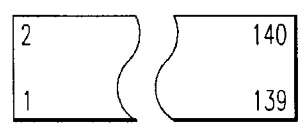
PIN#	SIGNAL	SH#	PIN#	SIGNAL	SH#	PIN#	SIGNAL	SH#	PIN#	SIGNAL	SH#	PIN#	SIGNAL	SH#	PIN#	SIGNAL	SH#	PIN#	SIGNAL	SH#
1	GND	20	21	RESET	19	41	PWROFF-	19	61	IOR-	03	81	SD5	06	101	SA5	11	121	DREQ6	11
2	VCC	20	22	LMICIN	10	42	PWRON-	19	62	MEMW-	03	82	SD6	06	102	SA6	11	122	DREQ7	11
3	IRRST-	15	23	IRD_C-	15	43	SUSRES	14	63	IRQ3	11	83	SD7	06	103	SA7	11	123	DACK0-	11
4	VIN	18	24	RTLNIN	10	44	AEN	03	64	IRQ4	11	84	SD8	06	104	SA8	11	124	DACK7-	11
5	GND	20	25	MIDITXD	10	45	SHUT-	14	65	IRQ5	11	85	SD9	06	105	SA9	11	125	IRBSY	15
6	TVDN	09	26	LINGND	10	46	ZEROWS-	03	66	IRQ6	11	86	SD10	06	106	SA10	11	126	DACK6-	11
7	NTSCR	09	27	MIDIRXD	10	47	TC	03	67	IRQ7	11	87	SD11	06	107	SA11	11	127	DACK5-	11
8	NTSCB	09	28	LFLNIN	10	48	IOCS16-	03	68	IRQ9	11	88	SD12	06	108	SA12	11	128	DACK1-	11
9	CSYNC	09	29	GND	20	49	GND	20	69	IRQ10	11	89	VCC	20	109	VCC	20	129	VCC3	20
10	LFLNOUT	10	30	VCC	20	50	VCC	20	70	IRQ11	11	90	GND	20	110	GND	20	130	GND	20
11	AENSND	13	31	12V	20	51	SPKR	03	71	IRQ14	11	91	SD13	06	111	SA13	11	131	LA23	11
12	LNOUTGND	10	32	PWRGOOD	19	52	REF-	03	72	IRQ15	11	92	SD14	06	112	SA14	11	132	DACK3-	11
13	IRRXD	15	33	AENCIA	13	53	IOCHRDY	03	73	IRQ12	11	93	SD15	06	113	SA15	11	133	LA21	11
14	RTLNOUT	10	34	NUM_LED-	14	54	BALE	03	74	SD0	06	94	SA0	11	114	SA16	11	134	LA22	11
15	NTSCG	09	35	PWR_LED-	14	55	MEMCS16-	03	75	SD1	06	95	SA1	11	115	DREQ5	11	135	LA19	11
16	IRTXD	15	36	SCR_LED-	14	56	IOW-	03	76	SD2	06	96	SA2	11	116	DREQ3	11	136	LA20	11
17	NTSC	09	37	CAP_LED-	14	57	SBHE-	03	77	SD3	06	97	SA3	11	117	DREQ0	11	137	LA17	11
18	MICGND	10	38	PBLB	18	58	MEMR-	03	78	SD4	06	98	SA4	11	118	DREQ1	11	138	LA18	11
19	VCC	20	39	VCC3	20	59	GND	20	79	GND	20	99	GND	20	119	GND	20	139	GND	20
20	GND	05	40	GND	20	60	VCC	20	80	VCC3	20	100	VCC	20	120	VCC	20	140	VCC	20

P150 IR MODULE

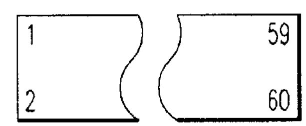
PIN#	SIGNAL	SH#
1	VCC	15
2	GND	15
3	IRTXD	15
4	IRRXD	15
5	IRBSY	15
6	IRD_C-	15
7	N.C.	15
8	IRRST-	15

PWB TOP SIDE

8	7
2	1



140 PIN .8MM BOARD TO BOARD



60 PIN .8MM BOARD TO BOARD

FIG. 30C

P21 IDE

PIN#	SIGNAL	SH#	PIN#	SIGNAL	SH#
1	HDRESET	16	23	HDIOW-	16
2	GND	16	24	GND	16
3	IDED7	16	25	HDIOR-	16
4	HDD8	16	26	GND	16
5	HDD6	16	27	N.C.	16
6	HDD9	16	28	N.C.	16
7	HDD5	16	29	N.C.	16
8	HDD10	16	30	GND	16
9	HDD4	16	31	IRQ14	16
10	HDD11	16	32	IOCS16-	16
11	HDD3	16	33	HDA1	16
12	HDD12	16	34	N.C.	16
13	HDD2	16	35	HDA0	16
14	HDD13	16	36	HDA2	16
15	HDD1	16	37	HDCS0-	16
16	HDD14	16	38	HDCS1-	16
17	HDD0	16	39	HDLED-	16
18	HDD15	16	40	GND	16
19	GND	16	41	VCC	16
20	N.C.	16	42	VCC	16
21	N.C.	16	43	GND	16
22	GND	16	44	VCC	16

44 PIN 2MM HEADER

P141 KEYSCAN BOARD

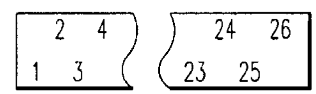
PIN#	SIGNAL	SH#	PIN#	SIGNAL	SH#	PIN#	SIGNAL	SH#
1	GND	22	21	EXTSLOW-	03	41	GND	05
2	GND	03	22	N.C.	04	42	GND	05
3	XD1	03	23	KGA20	04	43	N.C.	05
4	XD0	03	24	STANDBY-	04	44	N.C.	22
5	XD3	03	25	SELKBD	13	45	GND	05
6	XD2	03	26	EXUNIT-	04	46	GND	05
7	XD5	03	27	N.C.	04	47	BATA_DAT	05
8	XD4	03	28	IRQ11	22	48	BATB_DAT	05
9	XD7	03	29	CPUTMP	04	49	VCC	22
10	XD6	04	30	N.C.	18	50	VCC	22
11	VCC	05	31	GND	03	51	NUM_LED-	04
12	VCC	05	32	GND	03	52	FDD_LED-	04
13	KBCS2-	05	33	EXTPWR	03	53	PWR_LED-	04
14	8042CS-	05	34	VIN	03	54	HDD_LED-	03
15	IORB-	05	35	DIN_DAT	22	55	CAP_LED-	20
16	IOW-	05	36	DIN_CLK	03	56	TUR_LED-	04
17	SA2	05	37	GND	03	57	SCR_LED-	04
18	KEYHIT-	05	38	GND	03	58	PWRGOOD	04
19	IRQ1	04	39	KBDDAT	03	59	GND	04
20	IRQ12	04	40	KBDCLK	05	60	GND	22

P22 FLOPPY

PIN#	SIGNAL	SH#	PIN#	SIGNAL	SH#	PIN#	SIGNAL	SH#
1	VCC	16	10	MTRO-	16	19	GND	16
2	INDEX-	16	11	N.C.	16	20	TRKO-	16
3	VCC	16	12	DIR	16	21	GND	16
4	DRVO-	16	13	N.C.	16	22	WRPRT-	16
5	VCC	16	14	STEP-	16	23	GND	16
6	DSKCHG	16	15	GND	16	24	RDATA-	16
7	N.C.	16	16	WDATA-	16	25	GND	16
8	N.C.	16	17	GND	16	26	HDSEL	16
9	N.C.	16	18	WGATE-	16			

SOT23 MARKINGS	
2222	1PT
2907	2FE
DTA114	54E
DTC114	64
914BF	5D
BAT54	L4P
BAT54A	L42
BAT54S	L44
BAV70	A4W
5.6VZENER	8GH
BST82	02P

FIG. 30D



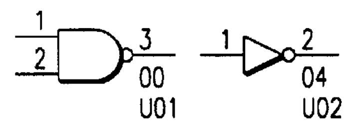
26 PIN 1MM FFC

NOTES : UNLESS OTHERWISE SPECIFIED :

1. ALL IC DEVICE TYPES ARE PREFIXED WITH SN74.
2. THE FOLLOWING PREFIX'S ARE ALWAY'S USED:
   T IS EQUAL TO "LS"
   AT IS EQUAL TO "ALS"
3. THE FOLLOWING PREFIX'S ARE USED ONLY WHEN INSUFFICIENT CHARACTERS ARE AVAILABLE:
   A IS EQUAL TO "ACT"
   B IS EQUAL TO "BCT"
   V IS EQUAL TO "AS"
   W IS EQUAL TO "AT" OR "ALS"
4. IC PACKAGE TYPE IS INDICATED BY THE FOLLOWING SUFFIX'S:

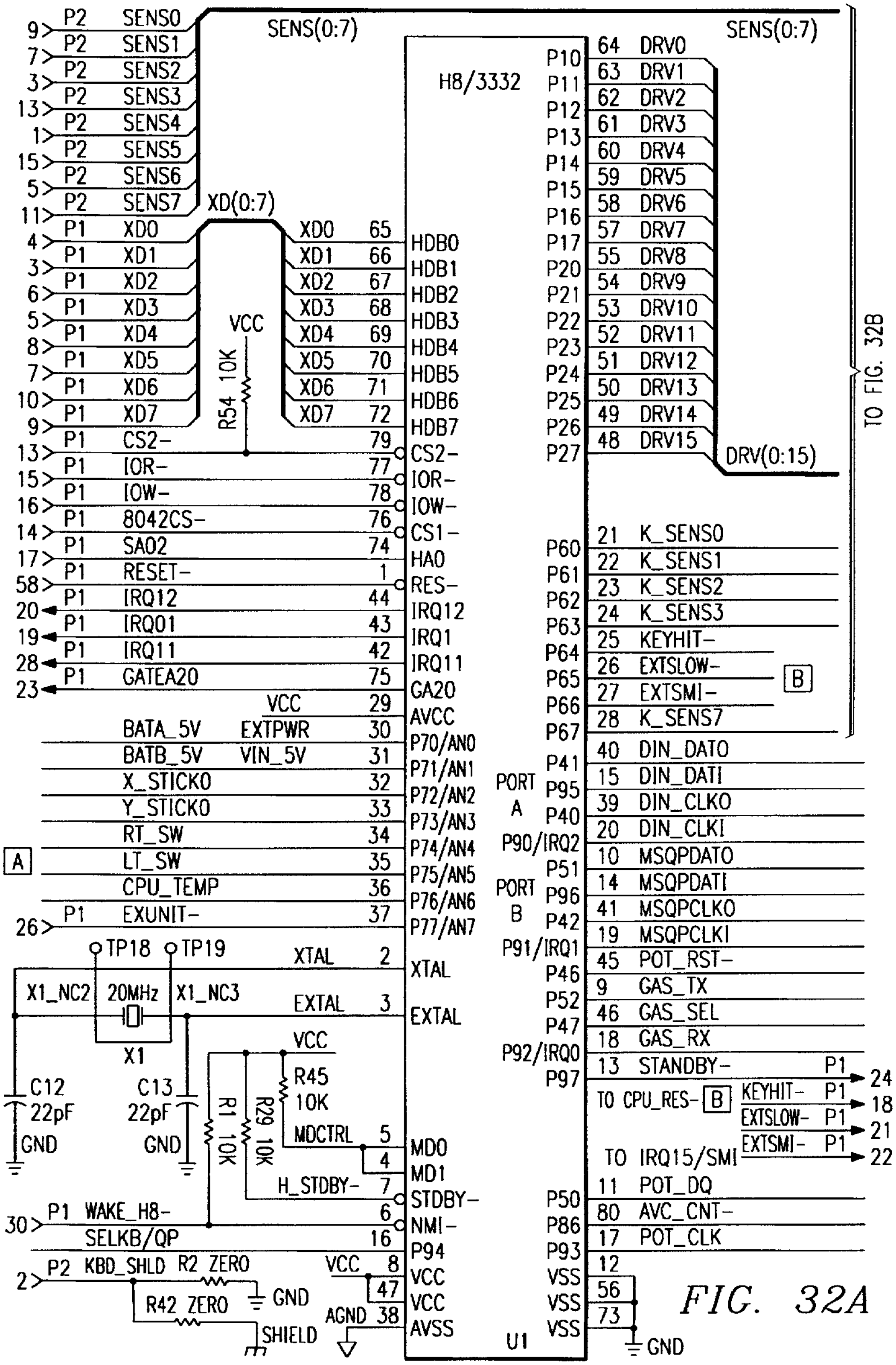
DUAL-IN-LINE, PLASTIC	= "N" OR BLANK
DUAL-IN-LINE, PLASTIC [WIDE]	= NW
DUAL-IN-LINE, CERAMIC	= J
DUAL-IN-LINE, CERAMIC [WIDE]	= JD
CHIP CARRIER, PLASTIC	= F
CHIP CARRIER IN A S.M. SCKT	= FF
CHIP CARRIER IN A PGA SCKT	= FX
CHIP CARRIER, CERAMIC [RECT]	= FE
CHIP CARRIER, CERAMIC [SQUARE]	= FH
FLAT PACKAGE, CERAMIC	= U
FLAT PACKAGE, CERAMIC [WIDE]	= W
GRID ARRAY, PLASTIC	= X
GRID ARRAY, PLASTIC [LIF SCKT]	= XL
GRID ARRAY, PLASTIC [ZIF SCKT]	= XZ
GRID ARRAY, CERAMIC	= Y
GRID ARRAY, CERAMIC [LIF SCKT]	= YL
GRID ARRAY, CERAMIC [ZIF SCKT]	= YZ
SINGLE-IN-LINE	= E,L,M,G
"SOIC", PLASTIC	= D
"SOIC", PLASTIC [WIDE]	= DW
"SOJ", PLASTIC, J LEADS	= R

5. VCC IS APPLIED TO PIN 8 OF ALL 8-PIN IC's, PIN 14 OF ALL 14-PIN IC's, PIN 16 OF ALL 16-PIN IC's, PIN 20 OF ALL 20-PIN IC's, ETC.
6. GROUND IS APPLIED TO PIN 4 OF ALL 8-PIN IC's, PIN 7 OF ALL 14-PIN IC's, PIN 8 OF ALL 16-PIN IC's, PIN 10 OF ALL 20-PIN IC's, ETC.
7. DEVICE TYPE, PIN NUMBERS, AND REFERENCE DESIGNATOR [LOCATION] OF GATES ARE SHOWN AS FOLLOWS:



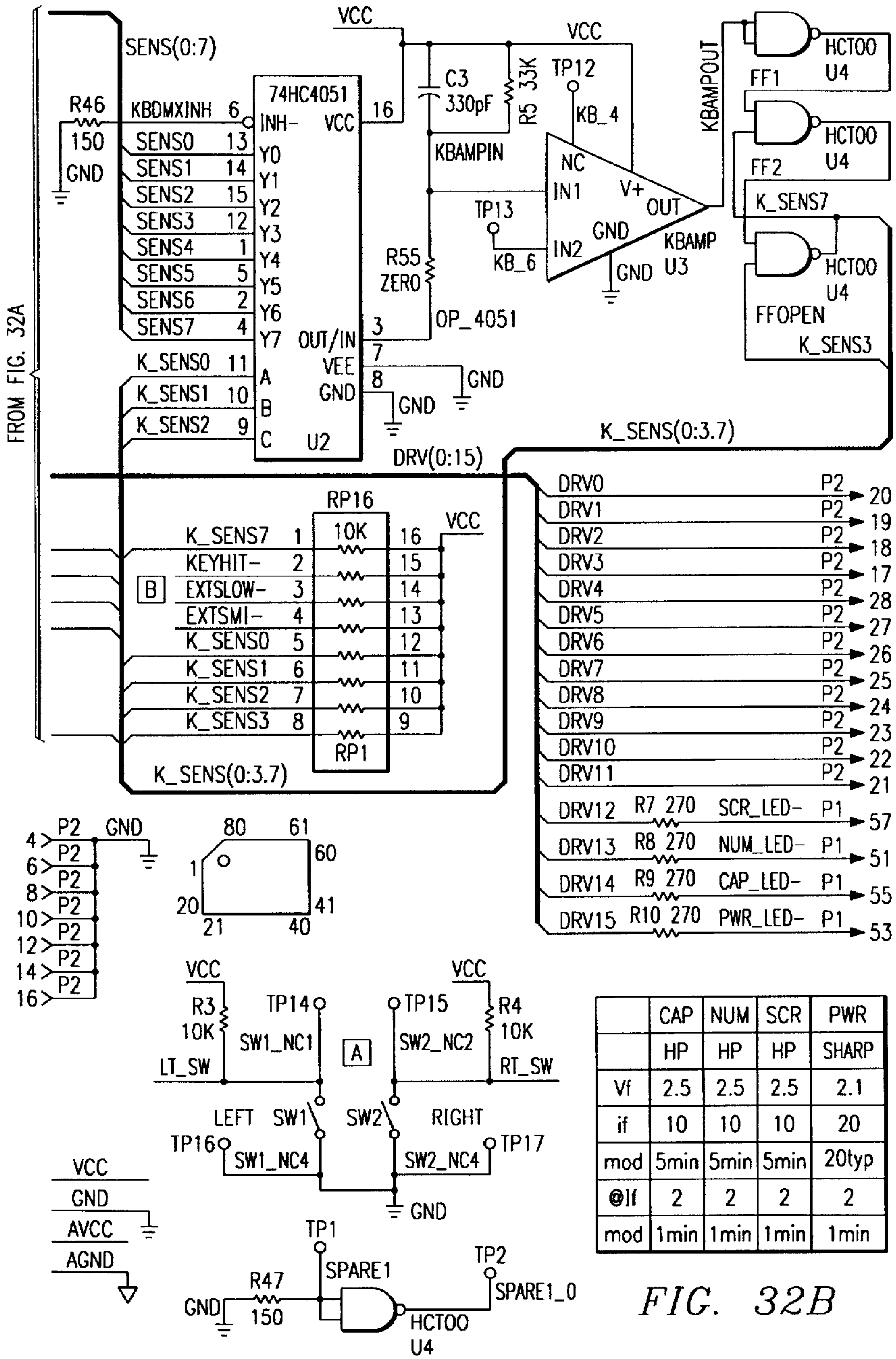
   00 AND 04 = DEVICE TYPES
   1, 2, AND 3 = PIN NUMBERS
   U01 AND U02 = REF. DESIGNATOR [LOCATION]
8. RESISTANCE VALUES ARE IN OHMS.
9. RESISTORS ARE 1/8 WATT, 5%.
10. CAPACITANCE VALUES ARE IN MICROFARADS.
11. CAPACITORS ARE 50V, 10%.
12. THIS COUPON WILL BE USED ON ALL COMMERICAL MULTILAYER BOARDS.

FIG. 31



H8/3332
SENS(0:7)
XD(0:7)
DRV(0:15)
TO FIG. 32B
VCC
R54 10K
R45 10K
R1 10K
R29 10K
20MHz
X1
X1_NC2
X1_NC3
TP18
TP19
C12 22pF
C13 22pF
GND
XTAL
EXTAL
MDCTRL
H_STDBY-
WAKE_H8-
SELKB/QP
KBD_SHLD
R2 ZERO
R42 ZERO
SHIELD
AGND
U1
PORT A
PORT B
BATA_5V
BATB_5V
EXTPWR
VIN_5V
X_STICK0
Y_STICK0
RT_SW
LT_SW
CPU_TEMP
EXUNIT-
K_SENS0
K_SENS1
K_SENS2
K_SENS3
KEYHIT-
EXTSLOW-
EXTSMI-
K_SENS7
DIN_DAT0
DIN_DATI
DIN_CLK0
DIN_CLKI
MSQPDAT0
MSQPDATI
MSQPCLK0
MSQPCLKI
POT_RST-
GAS_TX
GAS_SEL
GAS_RX
STANDBY-
TO CPU_RES-
TO IRQ15/SMI
POT_DQ
AVC_CNT-
POT_CLK

FIG. 32A



	CAP	NUM	SCR	PWR
	HP	HP	HP	SHARP
Vf	2.5	2.5	2.5	2.1
if	10	10	10	20
mod	5min	5min	5min	20typ
@If	2	2	2	2
mod	1min	1min	1min	1min

FIG. 32B

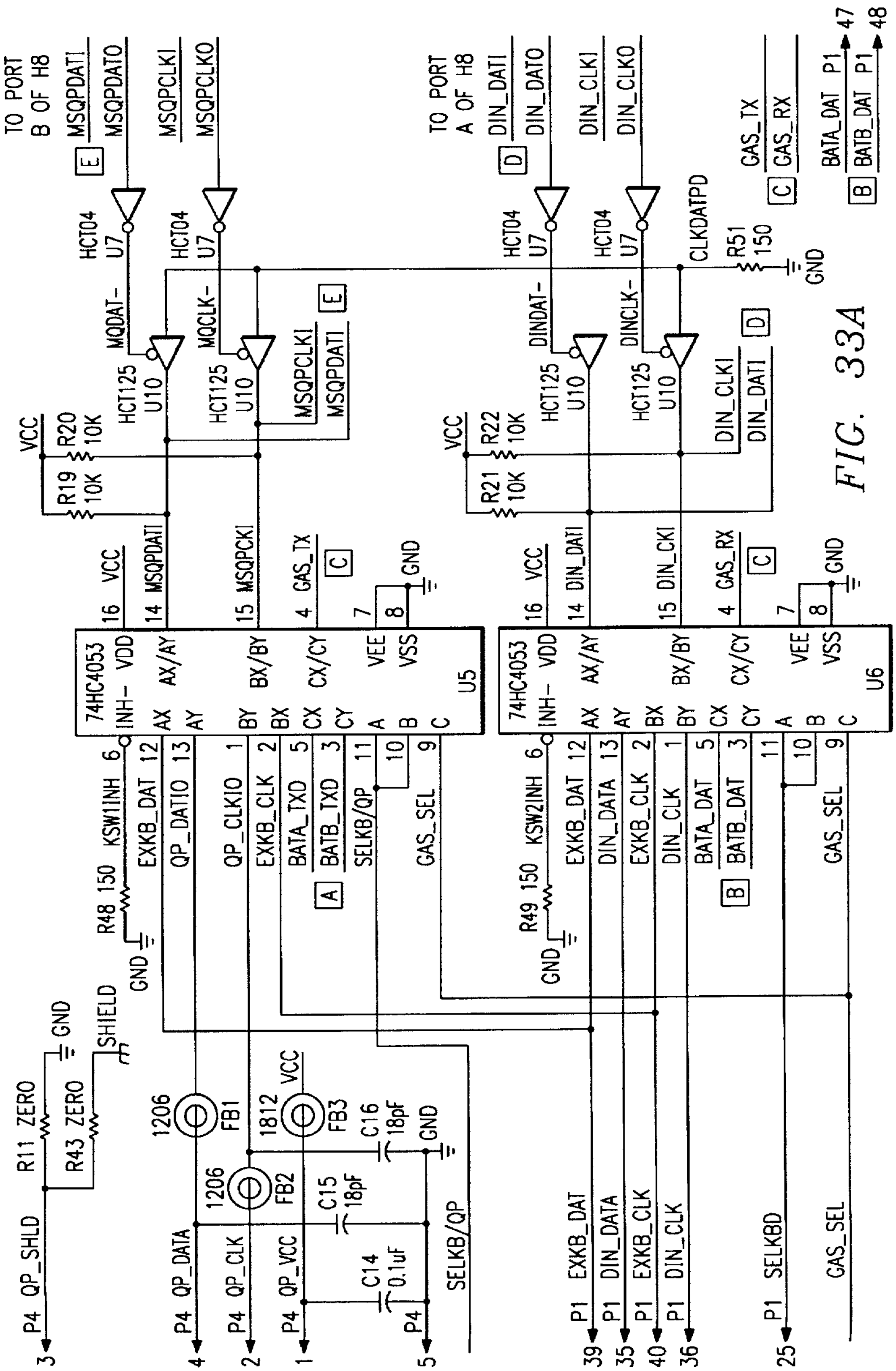

TO PORT B OF H8
MSQPDATI
MSQPDATO
MSQPCLKI
MSQPCLKO
TO PORT A OF H8
DIN_DATI
DIN_DATO
DIN_CLKI
DIN_CLKO
GAS_TX
GAS_RX
BATA_DAT P1 47
BATB_DAT P1 48
HCT04 U7
HCT125 U10
MQDAT-
MQCLK-
DINDAT-
DINCLK-
CLKDATPD
R51 150
GND
VCC
R19 10K
R20 10K
R21 10K
R22 10K
74HC4053
U5
U6
INH- VDD
AX/AY
BX/BY
CX/CY
VEE
VSS
KSW1INH
KSW2INH
EXKB_DAT
QP_DATIO
QP_CLKIO
EXKB_CLK
BATA_TXD
BATB_TXD
SELKB/QP
GAS_SEL
DIN_DATA
DIN_CLK
BATA_DAT
BATB_DAT
R48 150
R49 150
R11 ZERO
R43 ZERO
SHIELD
1206 FB1
1206 FB2
1812 FB3
C16 18pF
C15 18pF
C14 0.1uF
P4 QP_SHLD 3
P4 QP_DATA 4
P4 QP_CLK 2
P4 QP_VCC 1
P4 5
P1 EXKB_DAT 39
P1 DIN_DATA 35
P1 EXKB_CLK 40
P1 DIN_CLK 36
P1 SELKBD 25

FIG. 33A

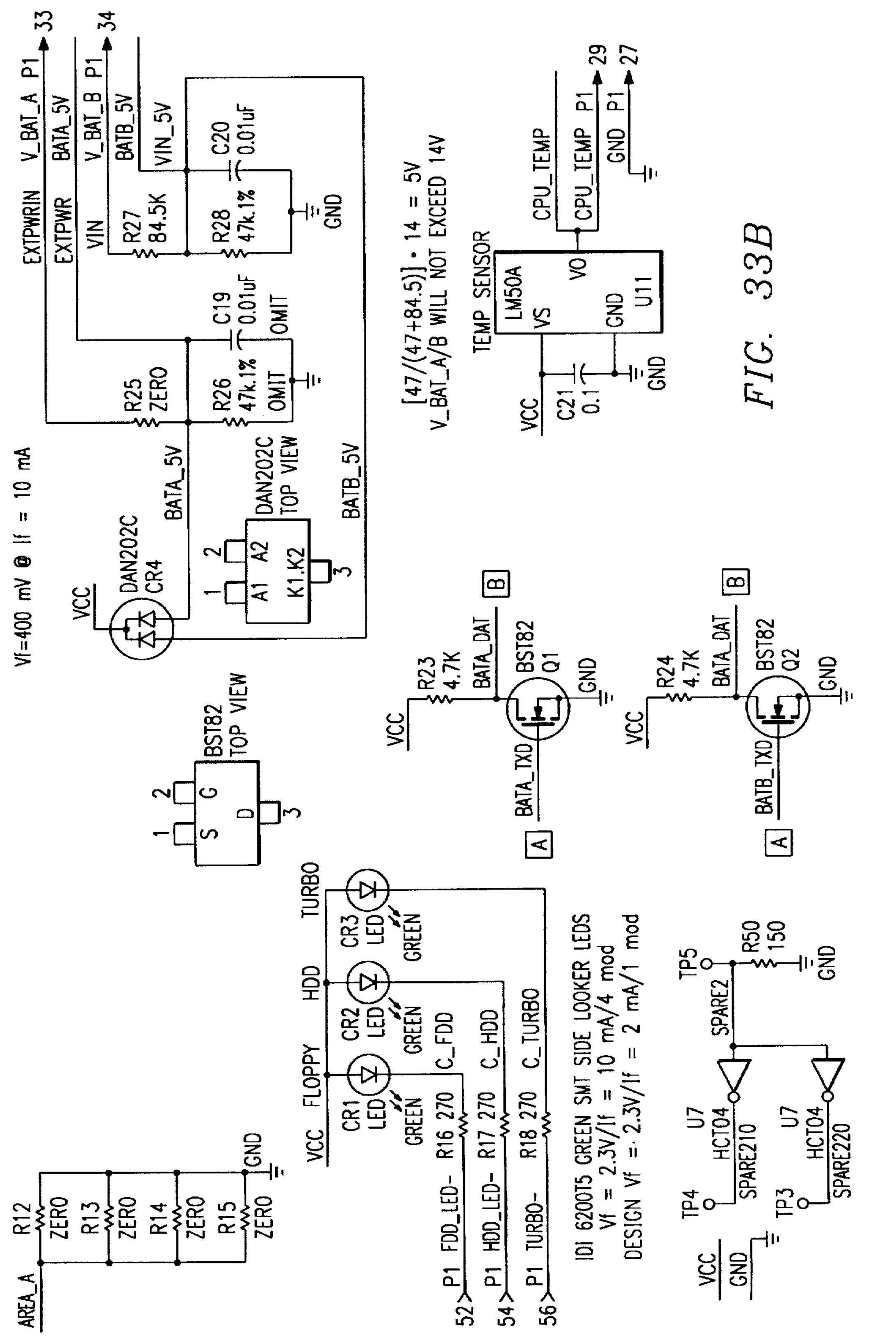
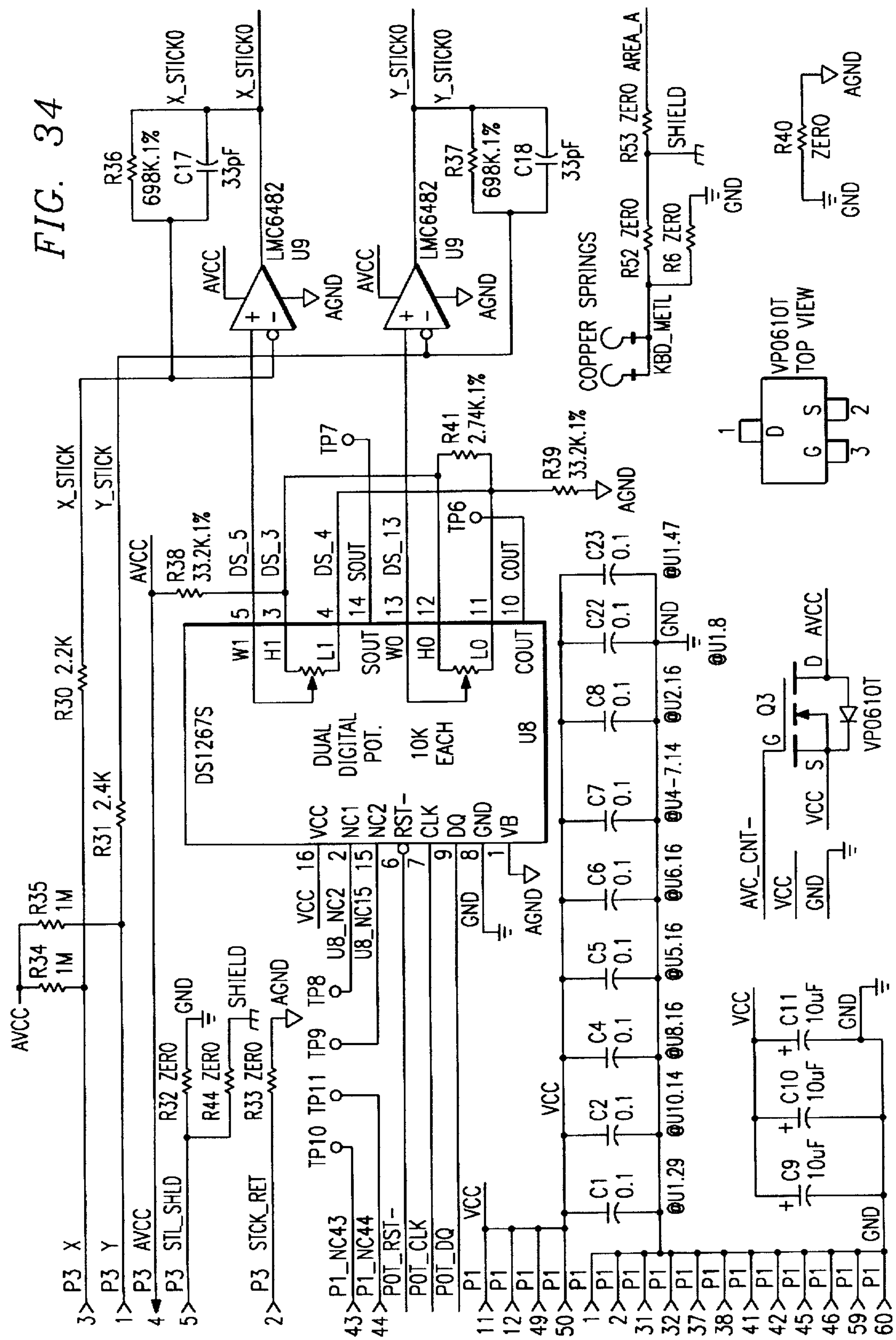

Vf=400 mV @ If = 10 mA
VCC
DAN202C
CR4
BATA_5V
DAN202C TOP VIEW
A1 A2
K1,K2
BATB_5V
EXTPWRIN V_BAT_A P1 33
EXTPWR BATA_5V
VIN V_BAT_B P1 34
BATB_5V
VIN_5V
R25 ZERO
R26 47k.1%
OMIT
C19 0.01uF
OMIT
R27 84.5K
R28 47k.1%
C20 0.01uF
GND
[47/(47+84.5)] • 14 = 5V
V_BAT_A/B WILL NOT EXCEED 14V
TEMP SENSOR
LM50A
VS
VO
GND
U11
CPU_TEMP
CPU_TEMP P1 29
GND P1 27
VCC
C21 0.1
BST82 TOP VIEW
S G D
R23 4.7K
BATA_DAT
BST82 Q1
BATA_TXD
R24 4.7K
BATA_DAT
BST82 Q2
BATB_TXD
AREA_A
R12 ZERO
R13 ZERO
R14 ZERO
R15 ZERO
VCC FLOPPY HDD TURBO
CR1 LED GREEN
CR2 LED GREEN
CR3 LED GREEN
R16 270 C_FDD
R17 270 C_HDD
R18 270 C_TURBO
P1 FDD_LED- 52
P1 HDD_LED- 54
P1 TURBO- 56
IDI 6200T5 GREEN SMT SIDE LOOKER LEDS
Vf = 2.3V/If = 10 mA/4 mod
DESIGN Vf = 2.3V/If = 2 mA/1 mod
TP5
R50 150
SPARE2
U7 HCT04
SPARE210
SPARE220
TP4
TP3
VCC
GND

FIG. 33B

FIG. 34



DS1267S
DUAL DIGITAL POT.
10K EACH
U8
LMC6482
U9
X_STICK0
Y_STICK0
X_STICK
Y_STICK
R36 698K.1%
C17 33pF
R37 698K.1%
C18 33pF
R30 2.2K
R31 2.4K
R34 1M
R35 1M
R38 33.2K.1%
R39 33.2K.1%
R41 2.74K.1%
R53 ZERO AREA_A
R52 ZERO
R6 ZERO
R40 ZERO
SHIELD
COPPER SPRINGS
KBD_METL
VP0610T TOP VIEW
AVC_CNT-
AVCC
AGND
VCC
GND

P1 8D2BD CONNECTOR - RECEPTACLE			
PIN#	SIGNAL	PIN#	SIGNAL
1	GND	2	GND
3	XD1	4	XD0
5	XD3	6	XD2
7	XD5	8	XD4
9	XD7	10	XD6
11	VCC	12	VCC
13	CS2-	14	8042CS-
15	IOR-	16	IOW-
17	SA02	18	KEYHIT-
19	IRQ1	20	IRQ12
21	CPU_RES-	22	EXTSMI-
23	GATEA20 (NU)	24	STANDBY-
25	SELKBD	26	EXUNIT-
27	CPU_TEMP_RET	28	IRQ11
29	CPU_TEMP	30	WAKE_H8-
31	GND	32	GND
33	EXTPWR	34	VIN
35	DIN_DATA (KBD/MSE)	36	DIN_CLK (KBD/MSE)
37	GND	38	GND
39	EXT_KBD_DATA	40	EXT_KBD_CLK
41	GND	42	GND
43	NC (KCLK OSC)	44	NC
45	GND	46	GND
47	BATA_DAT	48	BATB_DAT
49	VCC	50	VCC
51	NUM_LED-	52	FDD_LED-
53	POWER_LED-	54	HDD_LED-
55	CAP_LED-	56	TURBO_LED-
57	SCR_LED-	58	RESET-
59	GND	60	GND

SCH PG 02, 03, 04

1	59
2	60

P2 BROTHER KEYBOARD CONN			
PIN#	SIGNAL	PIN#	SIGNAL
1	SENS4	15	SENS5
2	SHIELD	16	GND
3	SENS2	17	DRV3
4	GND	18	DRV2
5	SENS6	19	DRV1
6	GND	20	DRV0
7	SENS1	21	DRV11
8	GND	22	DRV10
9	SENS0	23	DRV9
10	GND	24	DRV8
11	SENS7	25	DRV7
12	GND	26	DRV6
13	SENS3	27	DRV5
14	GND	28	DRV4

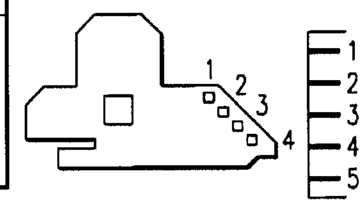
SCH PG 02

1	28

P3 STICK CONN	
PIN#	SIGNAL NAME
1	Y
2	GND
3	X
4	VCC
5	SHIELD

SCH PG 04

1	5



P4 QUICK PORT CONN	
PIN#	SIGNAL NAME
1	VCC
2	CLK
3	SHIELD
4	DATA
5	GND

SCH PG 03

1	5

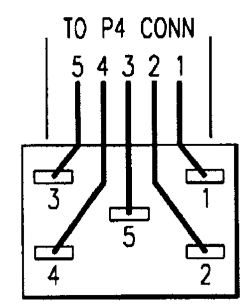


FIG. 35

NOTES : UNLESS OTHERWISE SPECIFIED :

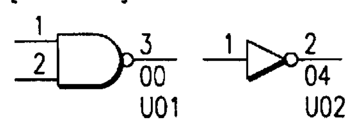
1. ALL IC DEVICE TYPES ARE PREFIXED WITH SN74.
2. THE FOLLOWING PREFIX'S ARE ALWAY'S USED:
   T IS EQUAL TO "LS"
   AT IS EQUAL TO "ALS"
3. THE FOLLOWING PREFIX'S ARE USED ONLY WHEN INSUFFICIENT CHARACTERS ARE AVAILABLE:
   A IS EQUAL TO "ACT"
   B IS EQUAL TO "BCT"
   V IS EQUAL TO "AS"
   W IS EQUAL TO "AT" OR "ALS"
4. IC PACKAGE TYPE IS INDICATED BY THE FOLLOWING SUFFIX'S:
   DUAL-IN-LINE, PLASTIC = "N" OR BLANK
   DUAL-IN-LINE, PLASTIC [WIDE] = NW
   DUAL-IN-LINE, CERAMIC = J
   DUAL-IN-LINE, CERAMIC [WIDE] = JD
   CHIP CARRIER, PLASTIC = F
   CHIP CARRIER IN A S.M. SCKT = FF
   CHIP CARRIER IN A PGA SCKT = FX
   CHIP CARRIER, CERAMIC [RECT] = FE
   CHIP CARRIER, CERAMIC [SQUARE] = FH
   FLAT PACKAGE, CERAMIC = U
   FLAT PACKAGE, CERAMIC [WIDE] = W
   GRID ARRAY, PLASTIC = X
   GRID ARRAY, PLASTIC [LIF SCKT] = XL
   GRID ARRAY, PLASTIC [ZIF SCKT] = XZ
   GRID ARRAY, CERAMIC = Y
   GRID ARRAY, CERAMIC [LIF SCKT] = YL
   GRID ARRAY, CERAMIC [ZIF SCKT] = YZ
   SINGLE-IN-LINE = E,L,M,G
   "SOIC", PLASTIC = D
   "SOIC", PLASTIC [WIDE] = DW
   "SOJ", PLASTIC, J LEADS = R

VCC'S NOT MARKED ON THE SCHEMATIC. ARE APPLIED AS FOLLOW:

power name	ref. designator	pin number
+5VSYS	U6	14
"	U33	14
"	U27	20
"	U37	20
"	U36	20
"	U22	14
VCC	U8	14
"	U11	14
SW_VCC	U10	14
+5VID	U28	16

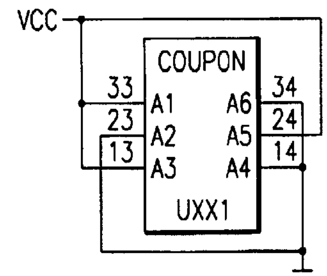
LAST REF.
DESIGNATORS USED:
R176 C183
L27 U40
CR9 SW3
TP5 Q26
QSC2 SP1

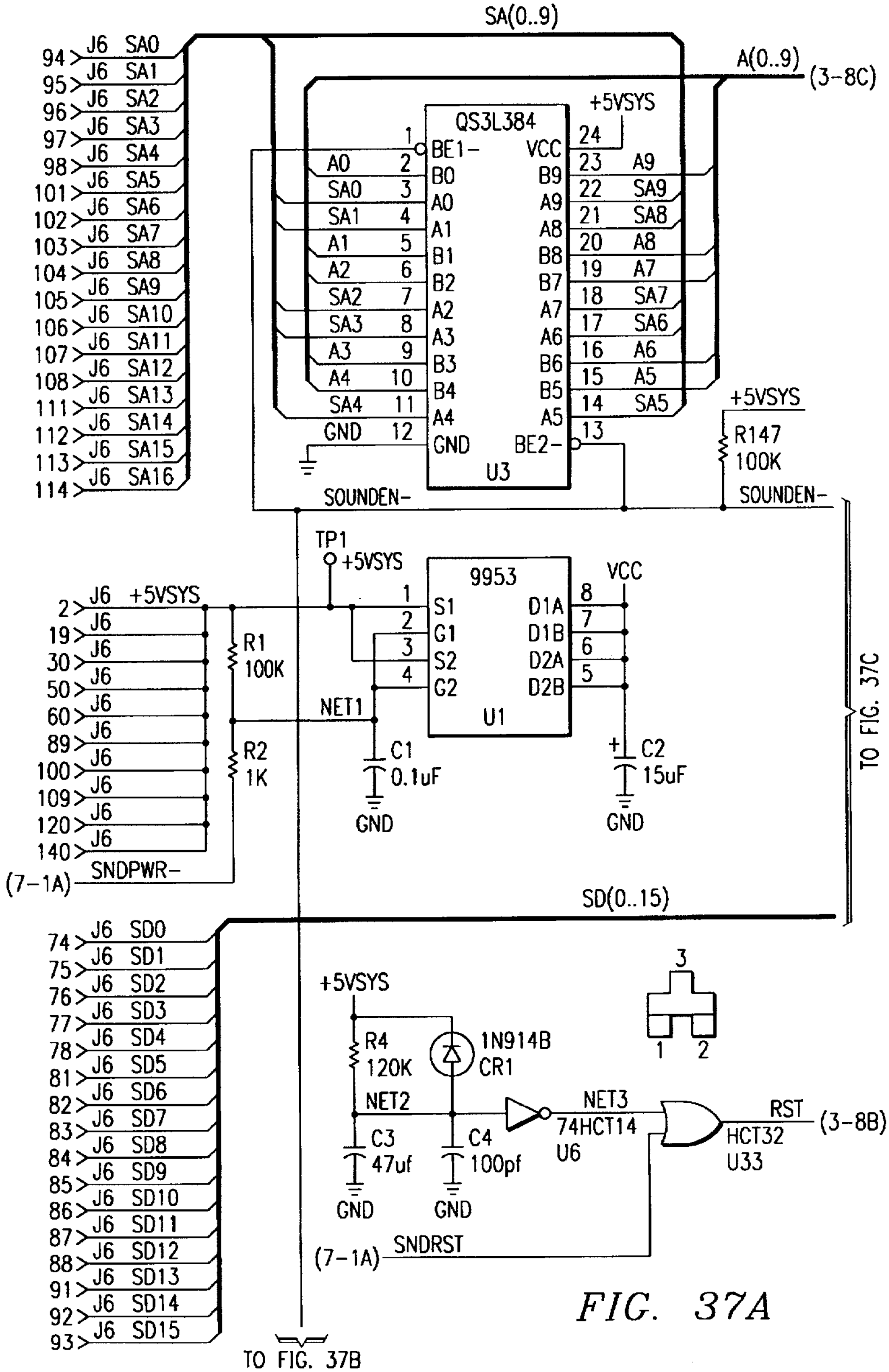
6. GROUND IS APPLIED TO PIN 4 OF ALL 8-PIN IC's, PIN 7 OF ALL 14-PIN IC's, PIN 8 OF ALL 16-PIN IC's, PIN 10 OF ALL 20-PIN IC's, ETC.
7. DEVICE TYPE, PIN NUMBERS, AND REFERENCE DESIGNATOR [LOCATION] OF GATES ARE SHOWN AS FOLLOWS:



   00 AND 04 = DEVICE TYPES
   1, 2, AND 3 = PIN NUMBERS
   U01 AND U02 = REF. DESIGNATOR [LOCATION]
8. RESISTANCE VALUES ARE IN OHMS.
9. RESISTORS ARE 1/8 WATT, 5%.
10. CAPACITANCE VALUES ARE IN MICROFARADS.
11. CAPACITORS ARE 50V, 10%.
12. THIS COUPON WILL BE USED ON ALL COMMERICAL MULTILAYER BOARDS.

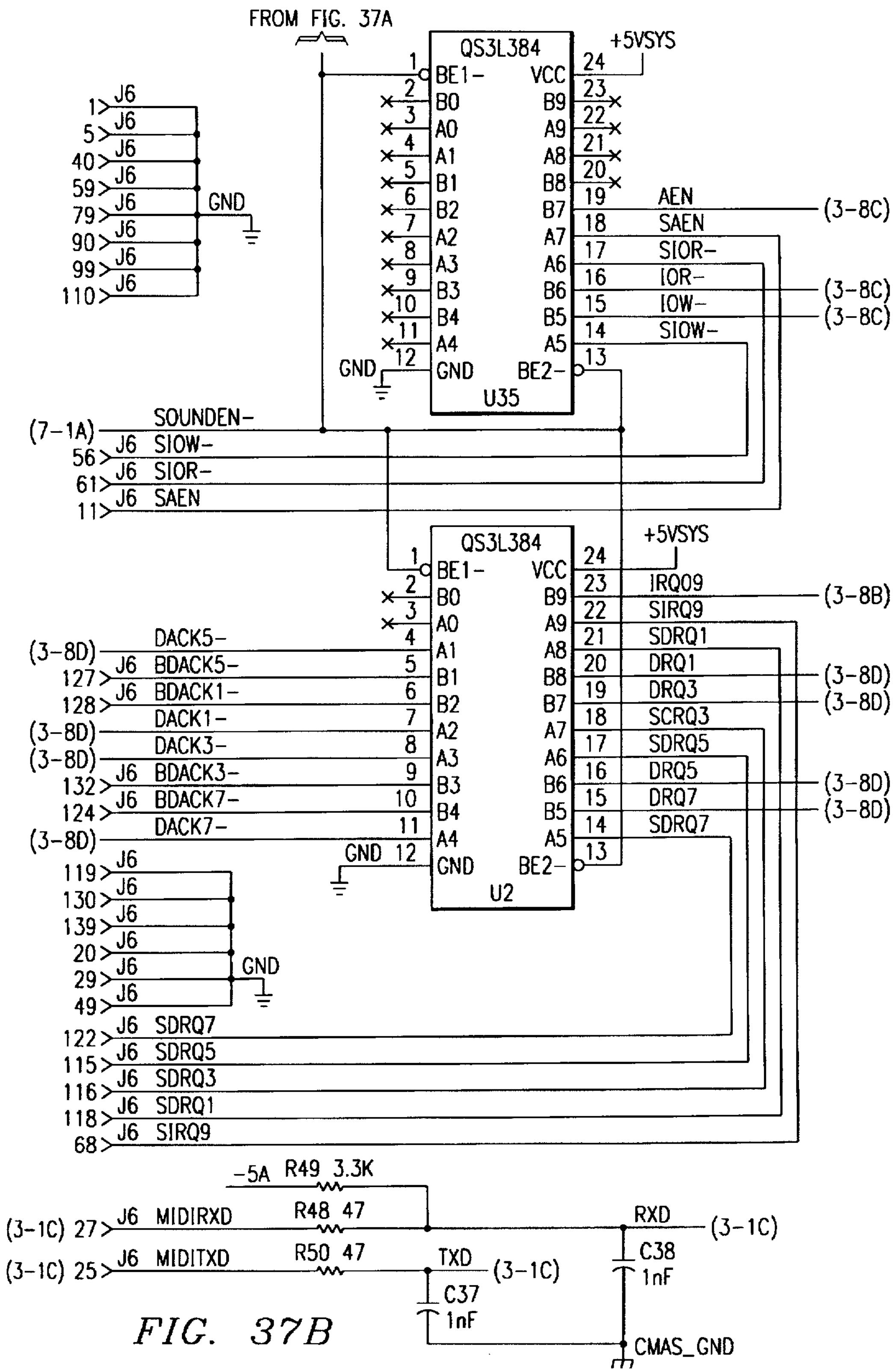
FIG. 36





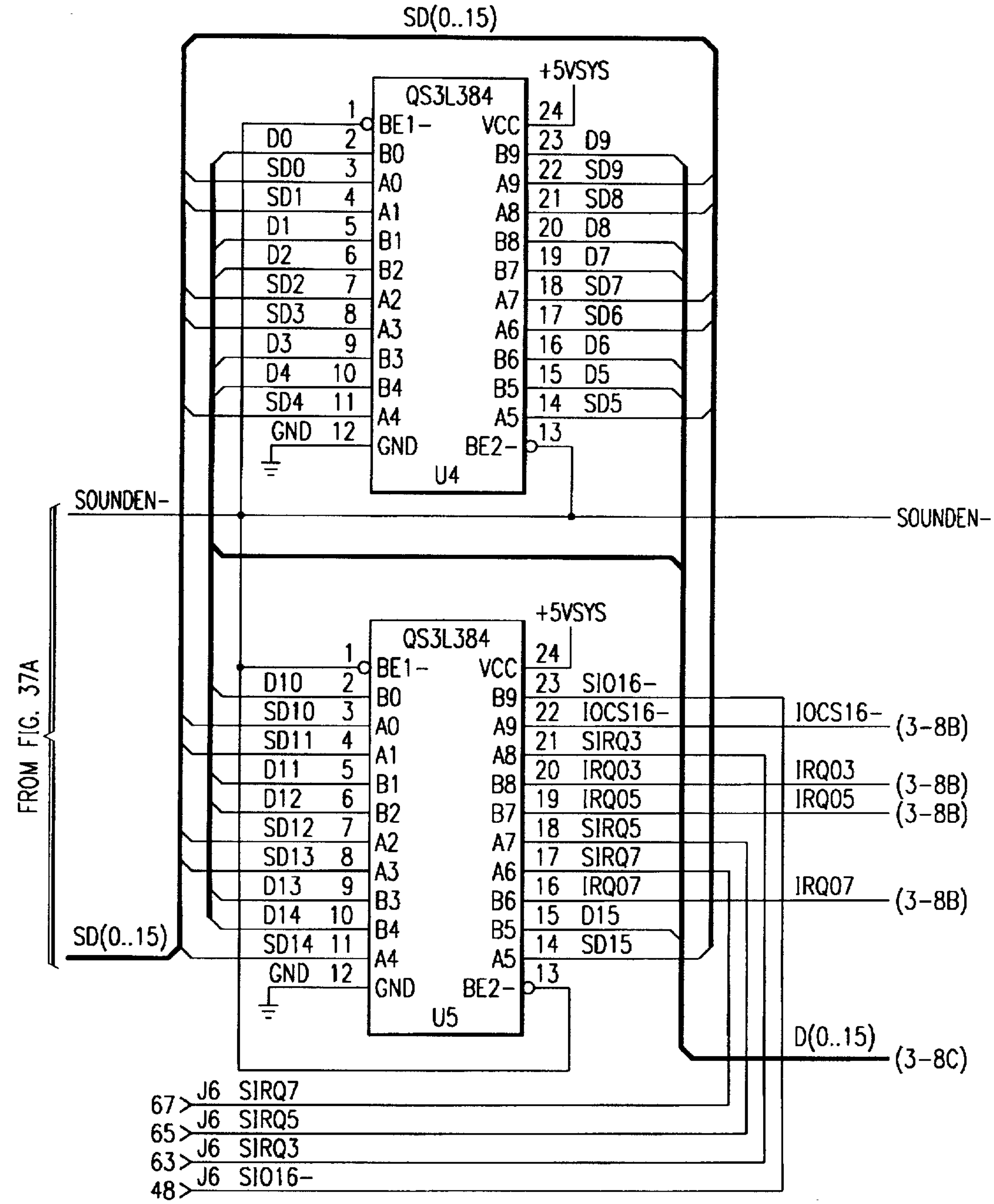
SA(0..9)
A(0..9)
(3-8C)
QS3L384
+5VSYS
VCC
BE1-
BE2-
U3
GND
SOUNDEN-
R147
100K
TP1
9953
U1
S1
G1
S2
G2
D1A
D1B
D2A
D2B
R1
100K
R2
1K
NET1
C1
0.1uF
C2
15uF
SNDPWR-
(7-1A)
SD(0..15)
TO FIG. 37C
R4
120K
1N914B
CR1
NET2
NET3
74HCT14
U6
C3
47uf
C4
100pf
HCT32
U33
RST
(3-8B)
SNDRST
TO FIG. 37B

FIG. 37A



FROM FIG. 37A
QS3L384
+5VSYS
BE1- VCC
B0 B9
A0 A9
A1 A8
B1 B8
B2 B7
A2 A7
A3 A6
B3 B6
B4 B5
A4 A5
GND BE2-
U35
AEN
SAEN
SIOR-
IOR-
IOW-
SIOW-
(3-8C)
GND
SOUNDEN-
(7-1A)
J6 SIOW-
J6 SIOR-
J6 SAEN
U2
IRQ09
SIRQ9
SDRQ1
DRQ1
DRQ3
SCRQ3
SDRQ5
DRQ5
DRQ7
SDRQ7
(3-8B)
(3-8D)
DACK5-
J6 BDACK5-
J6 BDACK1-
DACK1-
DACK3-
J6 BDACK3-
J6 BDACK7-
DACK7-
J6 SDRQ7
J6 SDRQ5
J6 SDRQ3
J6 SDRQ1
J6 SIRQ9
-5A R49 3.3K
J6 MIDIRXD
R48 47
J6 MIDITXD
R50 47
RXD
TXD
(3-1C)
C38 1nF
C37 1nF
CMAS_GND

FIG. 37B



SD(0..15)
QS3L384
+5VSYS
BE1-
VCC
D0
SD0
SD1
D1
D2
SD2
SD3
D3
D4
SD4
GND
D9
SD9
SD8
D8
D7
SD7
SD6
D6
D5
SD5
BE2-
U4
SOUNDEN-
SOUNDEN-
FROM FIG. 37A
QS3L384
+5VSYS
D10
SD10
SD11
D11
D12
SD12
SD13
D13
D14
SD14
GND
SIO16-
IOCS16-
SIRQ3
IRQ03
IRQ05
SIRQ5
SIRQ7
IRQ07
D15
SD15
BE2-
U5
IOCS16- (3-8B)
IRQ03 (3-8B)
IRQ05 (3-8B)
IRQ07 (3-8B)
D(0..15) (3-8C)
SD(0..15)
67 J6 SIRQ7
65 J6 SIRQ5
63 J6 SIRQ3
48 J6 SIO16-

FIG. 37C

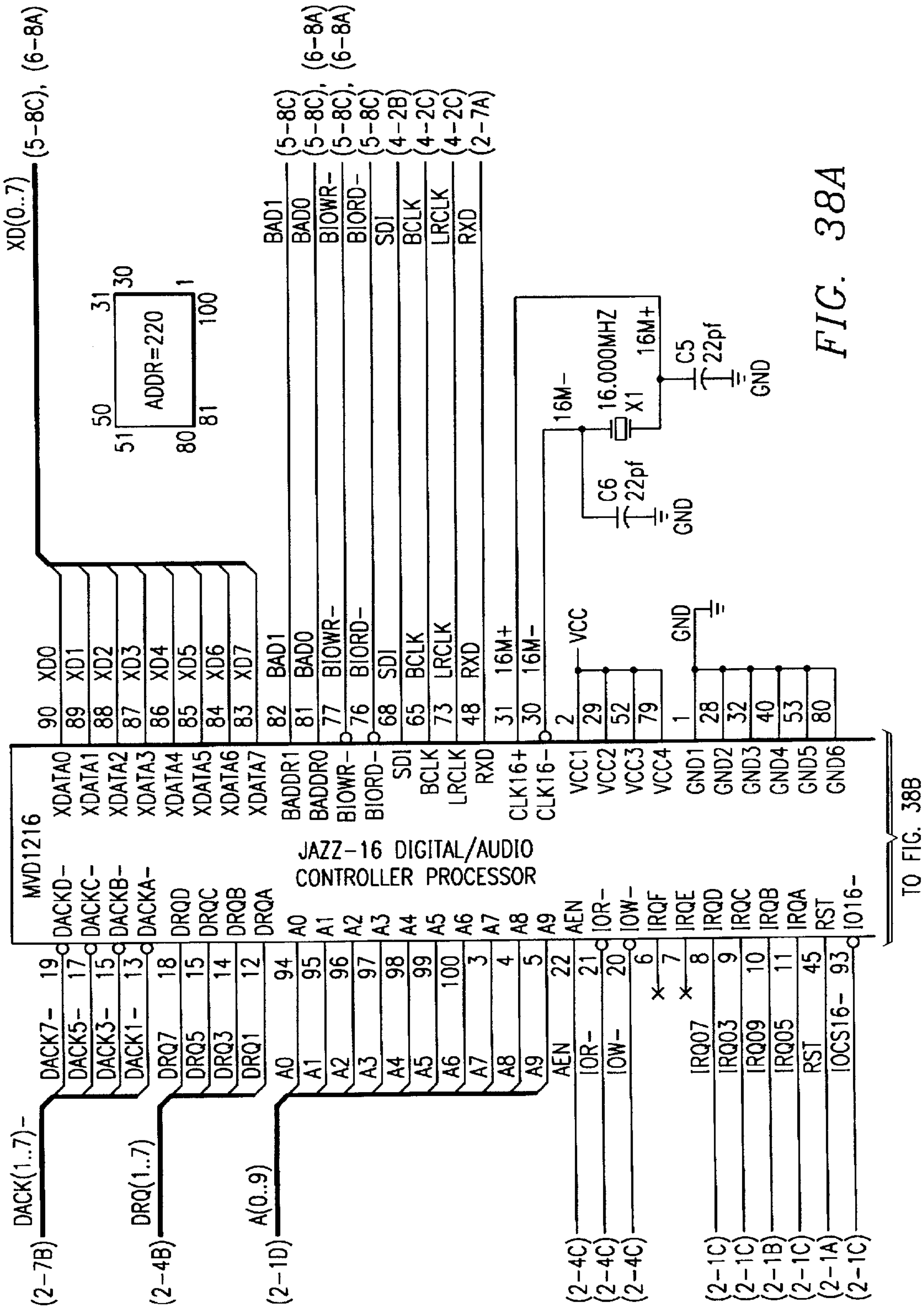

MVD1216
JAZZ-16 DIGITAL/AUDIO
CONTROLLER PROCESSOR
TO FIG. 38B
ADDR=220
XD(0..7)
DACK(1..7)-
DRQ(1..7)
A(0..9)
16.000MHZ
X1
C5
22pf
C6
22pf
GND
VCC

FIG. 38A

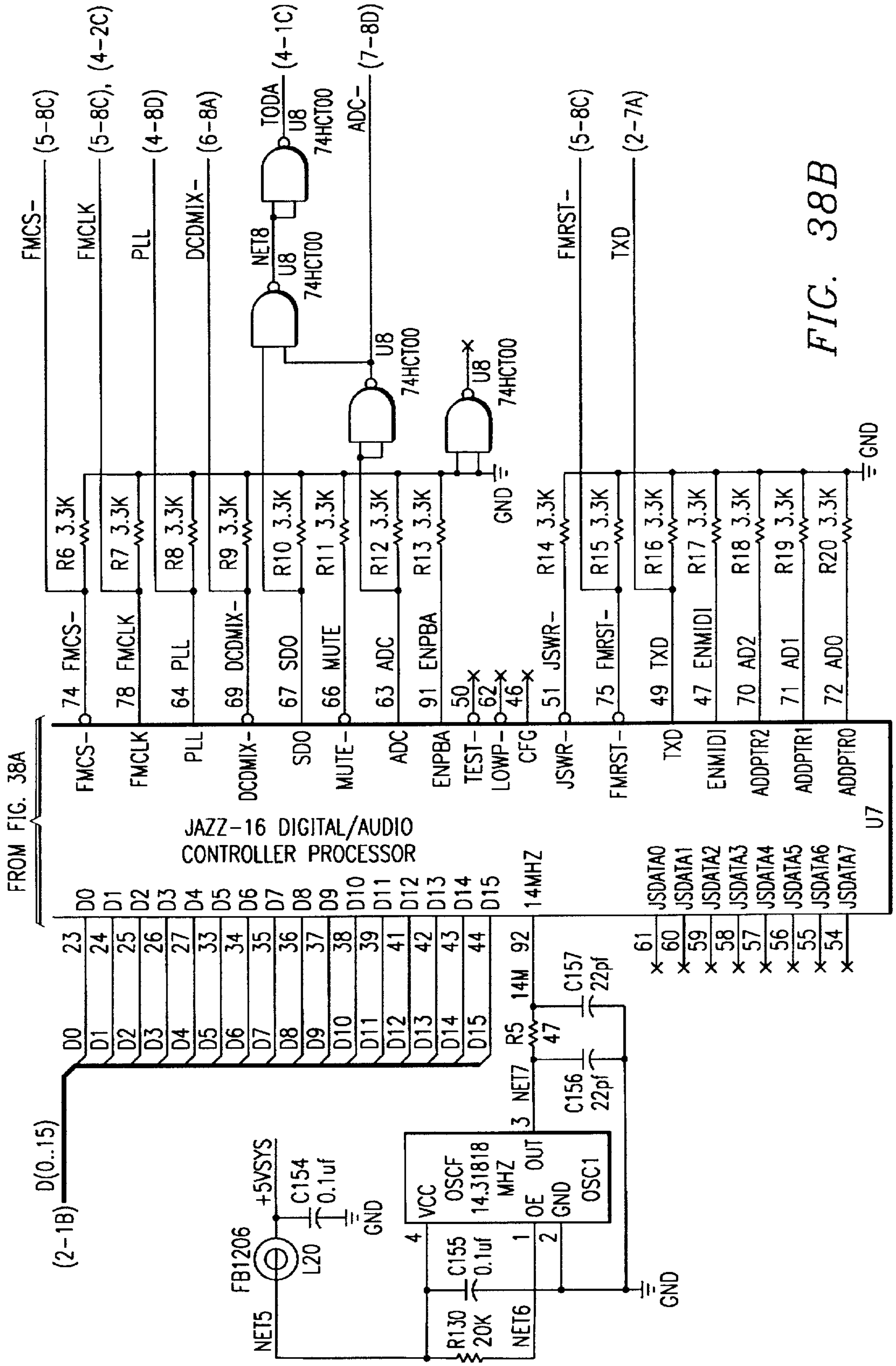

FROM FIG. 38A
JAZZ-16 DIGITAL/AUDIO
CONTROLLER PROCESSOR
U7
FMCS-
FMCLK
PLL
DCDMIX-
SDO
MUTE-
ADC
ENPBA
TEST-
LOWP-
CFG
JSWR-
FMRST-
TXD
ENMIDI
ADDPTR2
ADDPTR1
ADDPTR0
14MHZ
JSDATA0
JSDATA1
JSDATA2
JSDATA3
JSDATA4
JSDATA5
JSDATA6
JSDATA7
(5-8C)
(5-8C), (4-2C)
(4-8D)
(6-8A)
TODA
(4-1C)
ADC-
(7-8D)
(2-7A)
NET8
U8
74HCT00
GND
R6 3.3K
R7 3.3K
R8 3.3K
R9 3.3K
R10 3.3K
R11 3.3K
R12 3.3K
R13 3.3K
R14 3.3K
R15 3.3K
R16 3.3K
R17 3.3K
R18 3.3K
R19 3.3K
R20 3.3K
D(0..15)
(2-1B)
R5
47
14M
C157
22pf
C156
22pf
NET7
+5VSYS
C154
0.1uf
FB1206
L20
VCC
OSCF
14.31818
MHZ
OE
GND
OUT
OSC1
C155
0.1uf
R130
20K
NET5
NET6

FIG. 38B

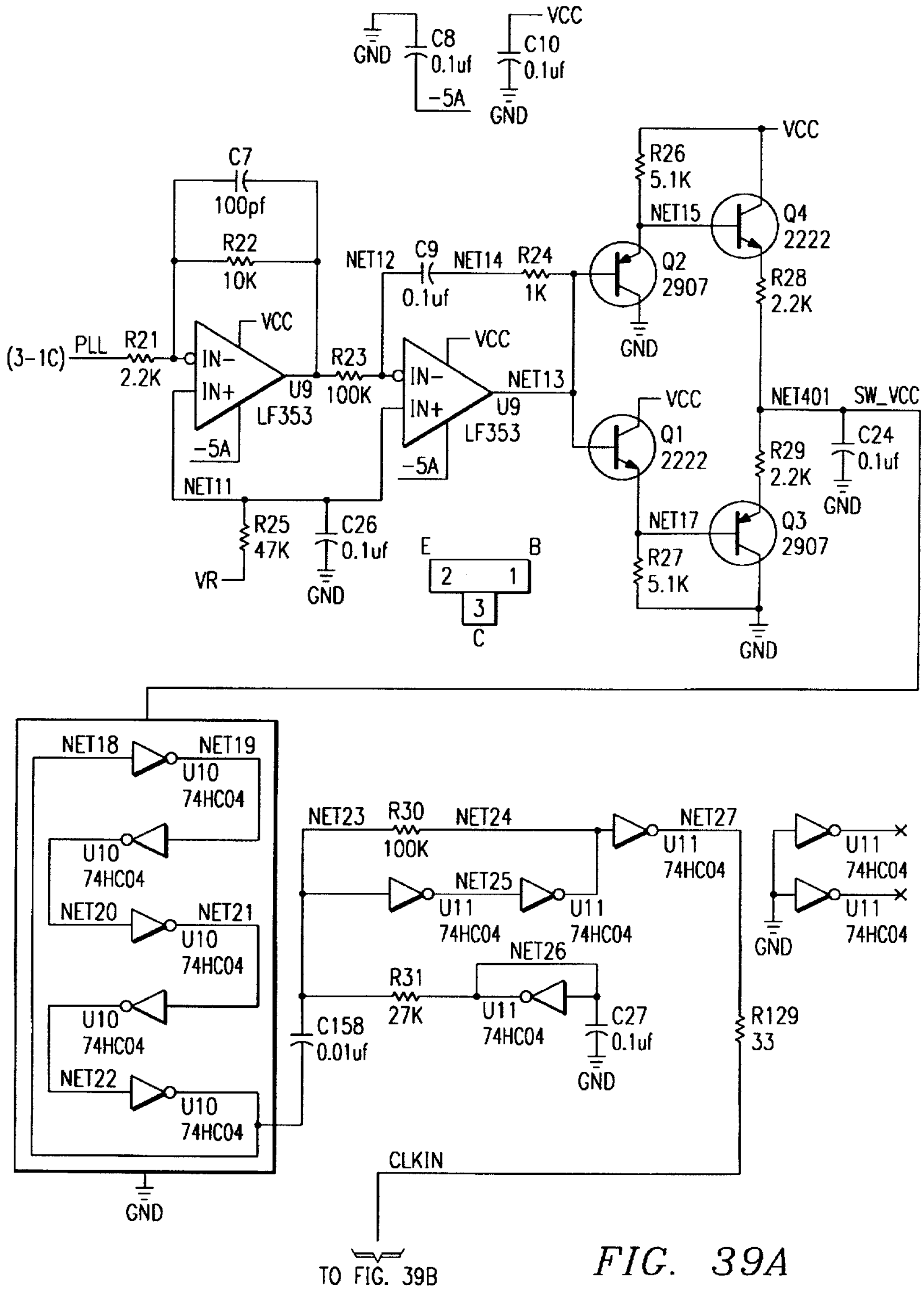

GND
C8
0.1uf
-5A
VCC
C10
0.1uf
GND
VCC
R26
5.1K
NET15
Q4
2222
C7
100pf
R22
10K
NET12
C9
0.1uf
NET14
R24
1K
Q2
2907
R28
2.2K
GND
VCC
(3-1C)
PLL
R21
2.2K
IN-
IN+
U9
LF353
-5A
R23
100K
VCC
IN-
IN+
NET13
U9
LF353
-5A
VCC
Q1
2222
NET401
SW_VCC
C24
0.1uf
R29
2.2K
GND
NET11
R25
47K
VR
C26
0.1uf
GND
E
B
2
1
3
C
NET17
Q3
2907
R27
5.1K
GND
NET18
NET19
U10
74HC04
U10
74HC04
NET20
NET21
U10
74HC04
U10
74HC04
NET22
U10
74HC04
GND
NET23
R30
100K
NET24
NET27
U11
74HC04
U11
74HC04
U11
74HC04
NET25
U11
74HC04
U11
74HC04
GND
NET26
R31
27K
U11
74HC04
C27
0.1uf
GND
C158
0.01uf
R129
33
CLKIN
TO FIG. 39B

FIG. 39A

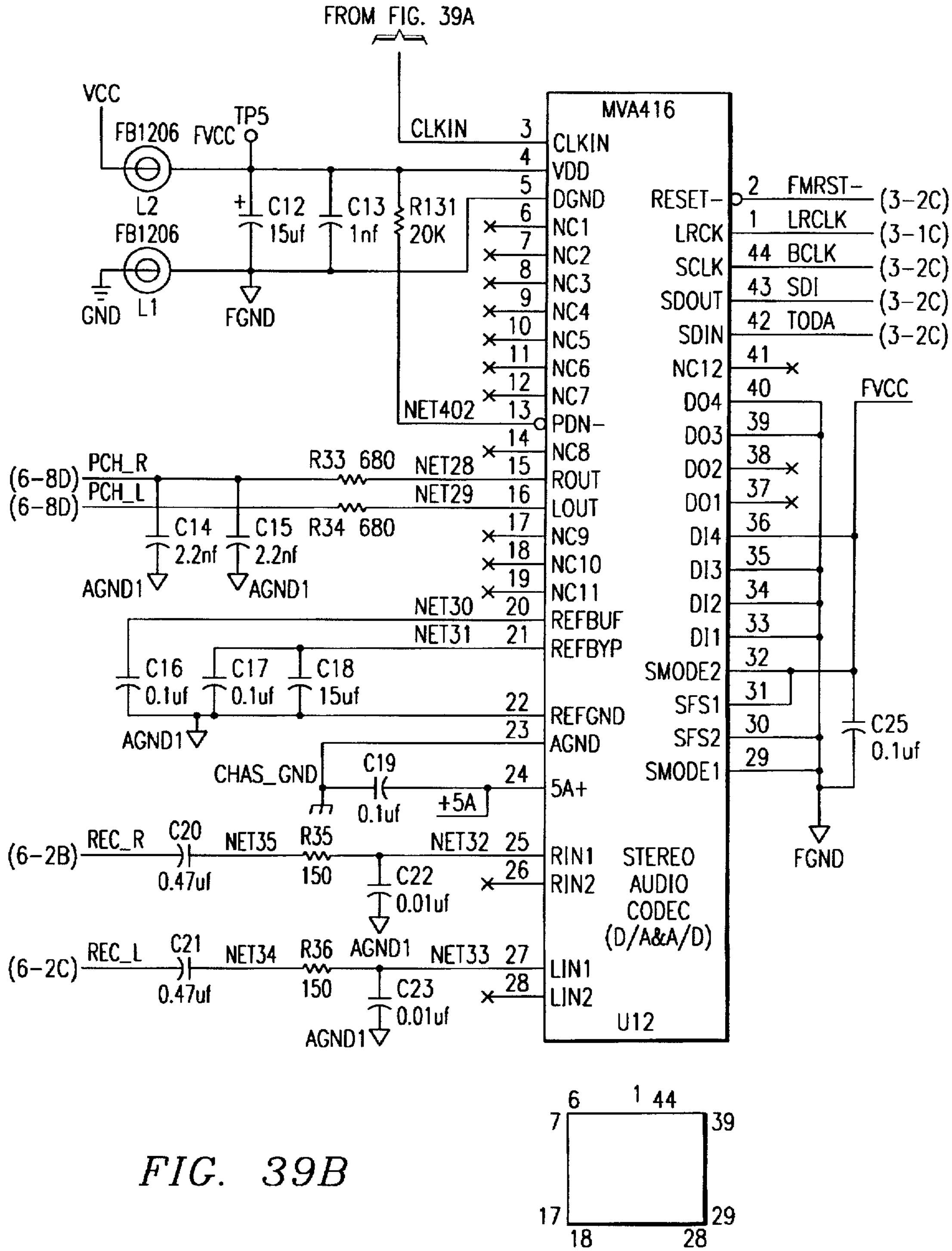

FROM FIG. 39A
VCC
FB1206
FVCC
TP5
L2
FB1206
GND
L1
FGND
C12
15uf
C13
1nf
R131
20K
CLKIN
MVA416
CLKIN
VDD
DGND
NC1
NC2
NC3
NC4
NC5
NC6
NC7
NET402
PDN-
NC8
PCH_R
PCH_L
(6-8D)
(6-8D)
R33 680
NET28
NET29
ROUT
LOUT
C14
2.2nf
C15
2.2nf
R34 680
AGND1
NC9
NC10
NC11
NET30
NET31
REFBUF
REFBYP
C16
0.1uf
C17
0.1uf
C18
15uf
REFGND
AGND
CHAS_GND
C19
0.1uf
+5A
5A+
(6-2B)
REC_R
C20
0.47uf
NET35
R35
150
NET32
RIN1
RIN2
C22
0.01uf
STEREO
AUDIO
CODEC
(D/A&A/D)
(6-2C)
REC_L
C21
0.47uf
NET34
R36
150
NET33
LIN1
LIN2
C23
0.01uf
U12
RESET-
FMRST-
(3-2C)
LRCK
LRCLK
(3-1C)
SCLK
BCLK
(3-2C)
SDOUT
SDI
(3-2C)
SDIN
TODA
(3-2C)
NC12
DO4
DO3
DO2
DO1
DI4
DI3
DI2
DI1
SMODE2
SFS1
SFS2
SMODE1
FVCC
C25
0.1uf
FGND

FIG. 39B

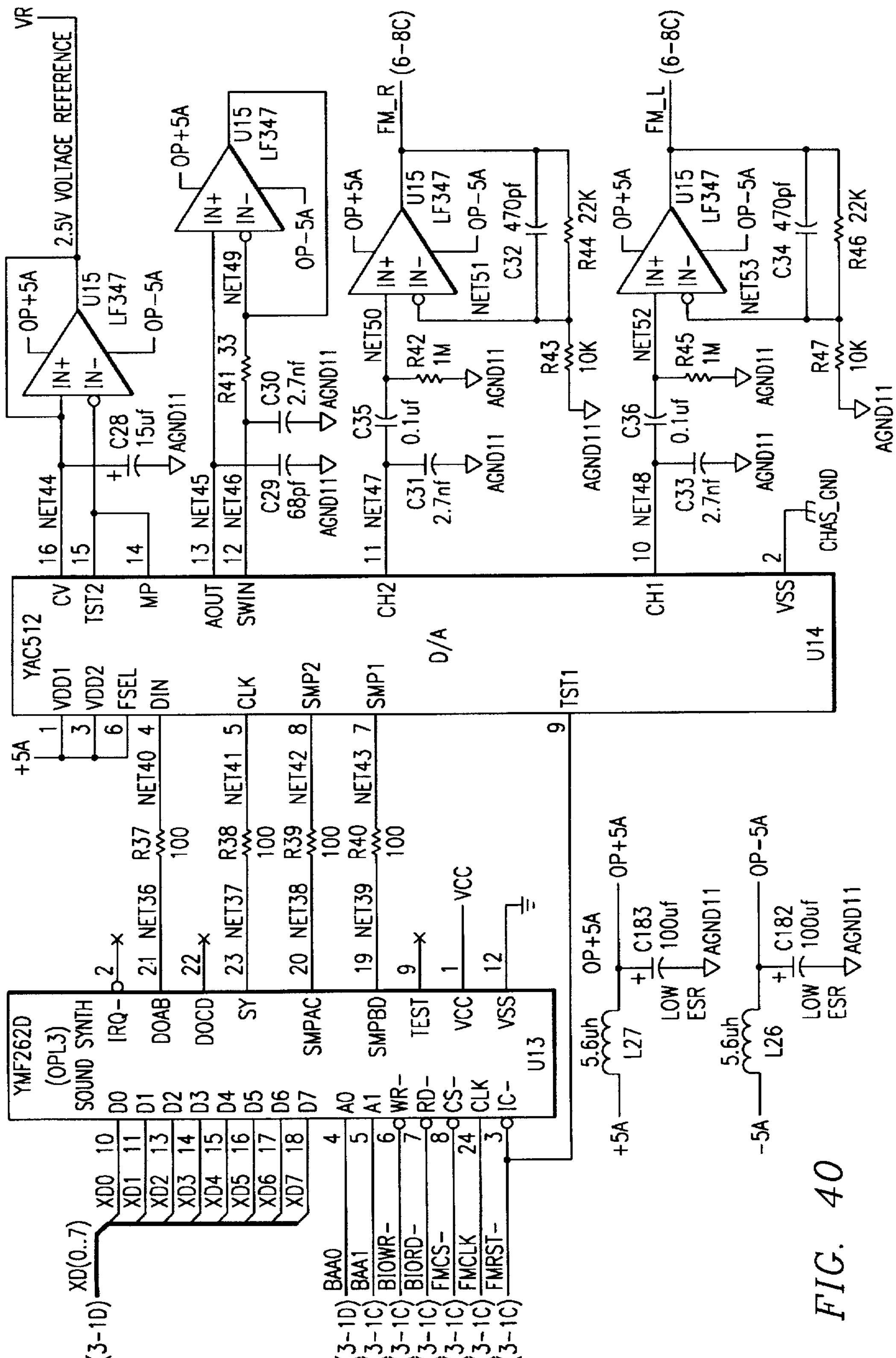

VR
2.5V VOLTAGE REFERENCE
U15 LF347
OP+5A
OP-5A
FM_R (6-8C)
FM_L (6-8C)
C32 470pf
R44 22K
C34 470pf
R46 22K
R42 1M
R43 10K
R45 1M
R47 10K
C28 15uf
C29 68pf
C30 2.7nf
C31 2.7nf
C33 2.7nf
C35 0.1uf
C36 0.1uf
R41 33
AGND11
CHAS_GND
YAC512
U14
D/A
CV TST2 MP AOUT SWIN CH2 CH1 VSS
VDD1 VDD2 FSEL DIN CLK SMP2 SMP1 TST1
NET44 NET45 NET46 NET47 NET48 NET49 NET50 NET51 NET52 NET53
NET36 NET37 NET38 NET39 NET40 NET41 NET42 NET43
R37 100
R38 100
R39 100
R40 100
+5A
VCC
YMF262D (OPL3) SOUND SYNTH
U13
IRQ- DOAB DOCD SY SMPAC SMPBD TEST VCC VSS
D0 D1 D2 D3 D4 D5 D6 D7 A0 A1 WR- RD- CS- CLK IC-
XD0 XD1 XD2 XD3 XD4 XD5 XD6 XD7
XD(0..7) (3-1D)
BAA0 (3-1D)
BAA1 (3-1C)
BIOWR- (3-1C)
BIORD- (3-1C)
FMCS- (3-1C)
FMCLK (3-1C)
FMRST- (3-1C)
L27 5.6uh
L26 5.6uh
C183 100uf LOW ESR
C182 100uf LOW ESR
-5A

FIG. 40

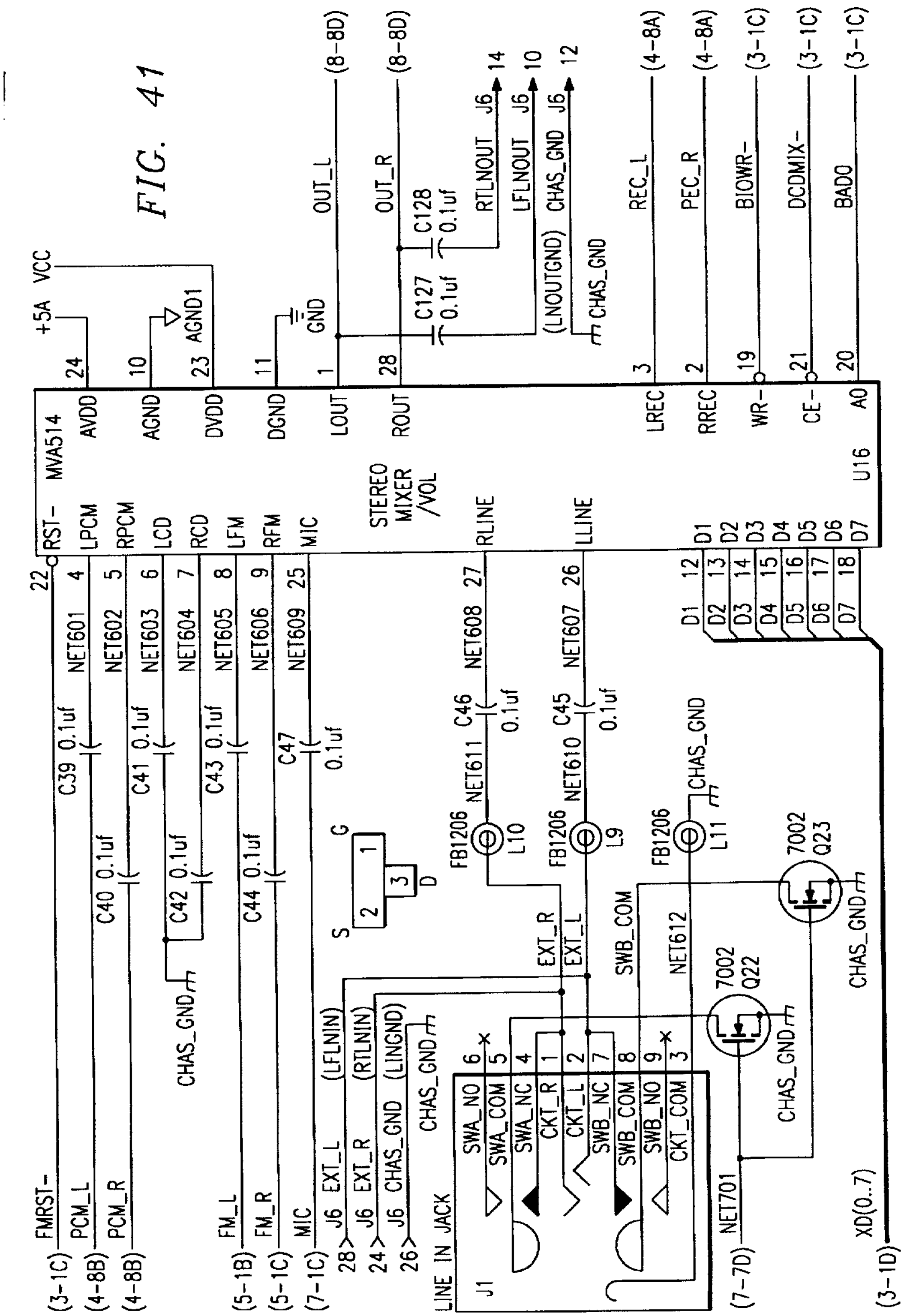

MVA514
U16
STEREO MIXER /VOL
AVDD
AGND
DVDD
DGND
LOUT
ROUT
LREC
RREC
WR-
CE-
A0
RST-
LPCM
RPCM
LCD
RCD
LFM
RFM
MIC
RLINE
LLINE
D1
D2
D3
D4
D5
D6
D7
+5A
VCC
AGND1
GND
OUT_L
OUT_R
C127 0.1uf
C128 0.1uf
RTLNOUT J6 14
LFLNOUT J6 10
CHAS_GND J6 12
(LNOUTGND)
CHAS_GND
REC_L
PEC_R
BIOWR-
DCDMIX-
BADO
(8-8D)
(4-8A)
(3-1C)
FMRST-
PCM_L
PCM_R
FM_L
FM_R
MIC
J6 EXT_L
J6 EXT_R
J6 CHAS_GND
(LFLNIN)
(RTLNIN)
(LINGND)
NET601
NET602
NET603
NET604
NET605
NET606
NET609
NET608
NET607
NET610
NET611
NET612
NET701
C39 0.1uf
C40 0.1uf
C41 0.1uf
C42 0.1uf
C43 0.1uf
C44 0.1uf
C47 0.1uf
C45 0.1uf
C46 0.1uf
FB1206
L9
L10
L11
EXT_R
EXT_L
SWB_COM
7002 Q22
7002 Q23
LINE IN JACK
J1
SWA_NO
SWA_COM
SWA_NC
CKT_R
CKT_L
SWB_NC
SWB_COM
SWB_NO
CKT_COM
XD(0..7)
(3-1D)
(7-7D)

FIG. 41

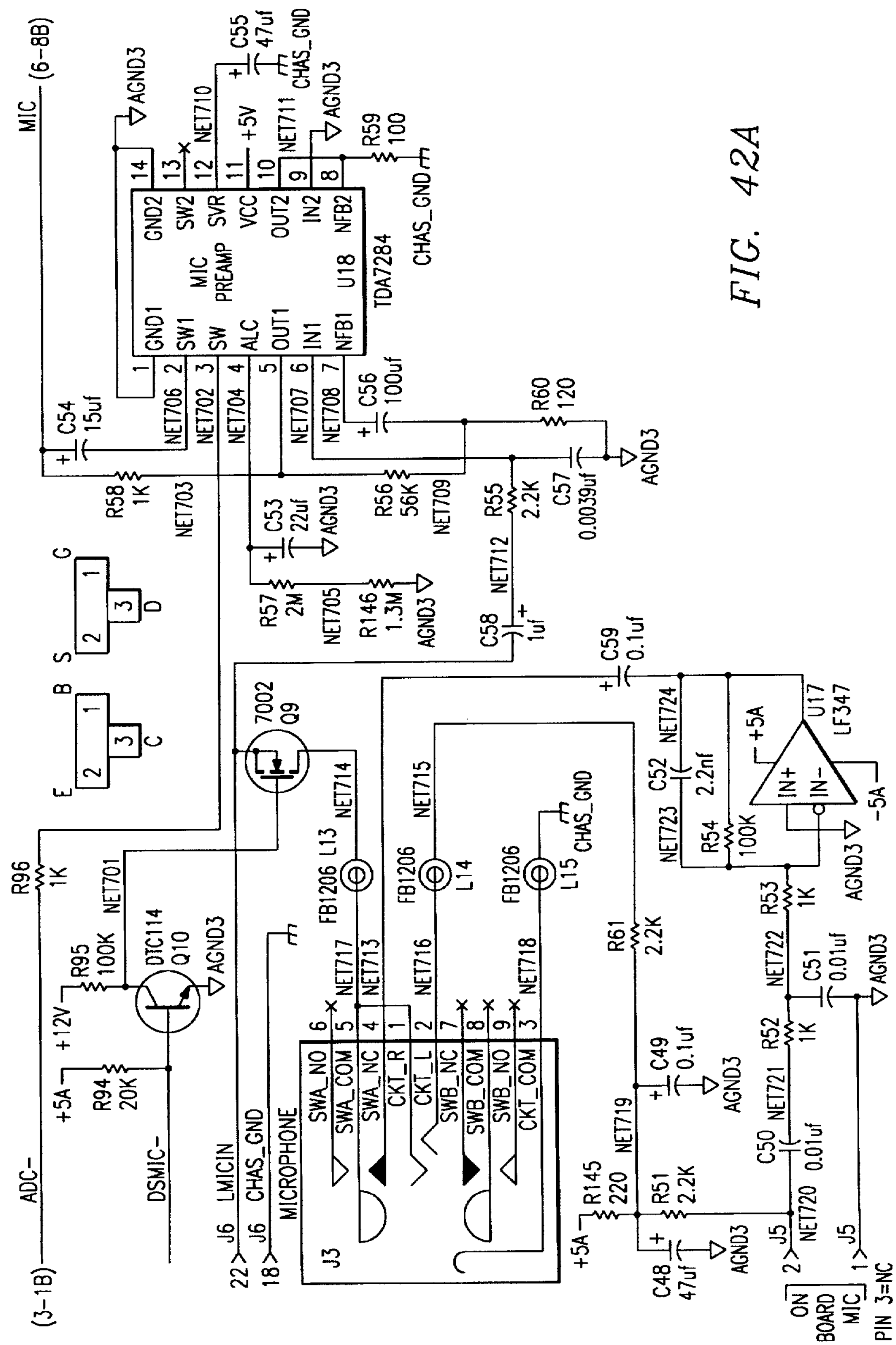

MIC (6-8B)
(3-1B) ADC-
AGND3
C55 47uf
CHAS_GND
NET710
+5V
NET711
R59 100
MIC PREAMP
U18 TDA7284
GND1 SW1 SW ALC OUT1 IN1 NFB1
GND2 SW2 SVR VCC OUT2 IN2 NFB2
C54 15uf
NET706
NET702
NET704
NET707
NET708
C56 100uf
R60 120
R58 1K
NET703
R56 56K
NET709
R55 2.2K
C57 0.0039uf
C53 22uf
R57 2M
NET705
R146 1.3M
NET712
C58 1uf
C59 0.1uf
NET724
U17 LF347
+5A
-5A
IN+ IN-
C52 2.2nf
NET723
R54 100K
R53 1K
NET722
C51 0.01uf
R52 1K
NET721
C50 0.01uf
NET720
7002 Q9
NET714
NET715
FB1206 L13
FB1206 L14
FB1206 L15
R96 1K
NET701
R95 100K
DTC114 Q10
+12V
+5A
R94 20K
DSMIC-
NET717
NET713
NET716
NET718
R61 2.2K
C49 0.1uf
NET719
R145 220
R51 2.2K
C48 47uf
J6 LMICIN
J6 CHAS_GND
MICROPHONE
J3
SWA_NO SWA_COM SWA_NC CKT_R CKT_L SWB_NC SWB_COM SWB_NO CKT_COM
J5
ON BOARD MIC
PIN 3=NC
E B C
S G D

FIG. 42A

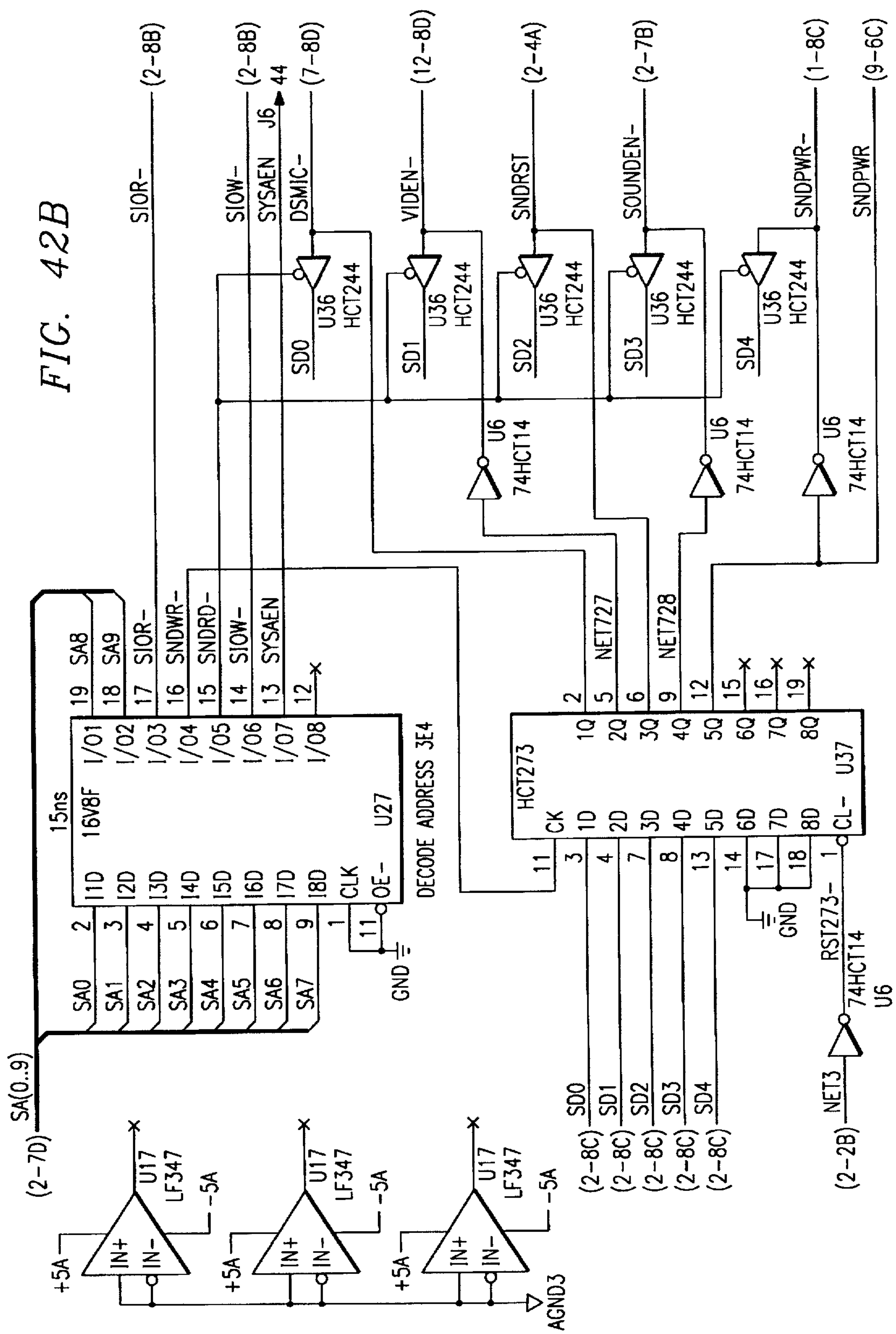
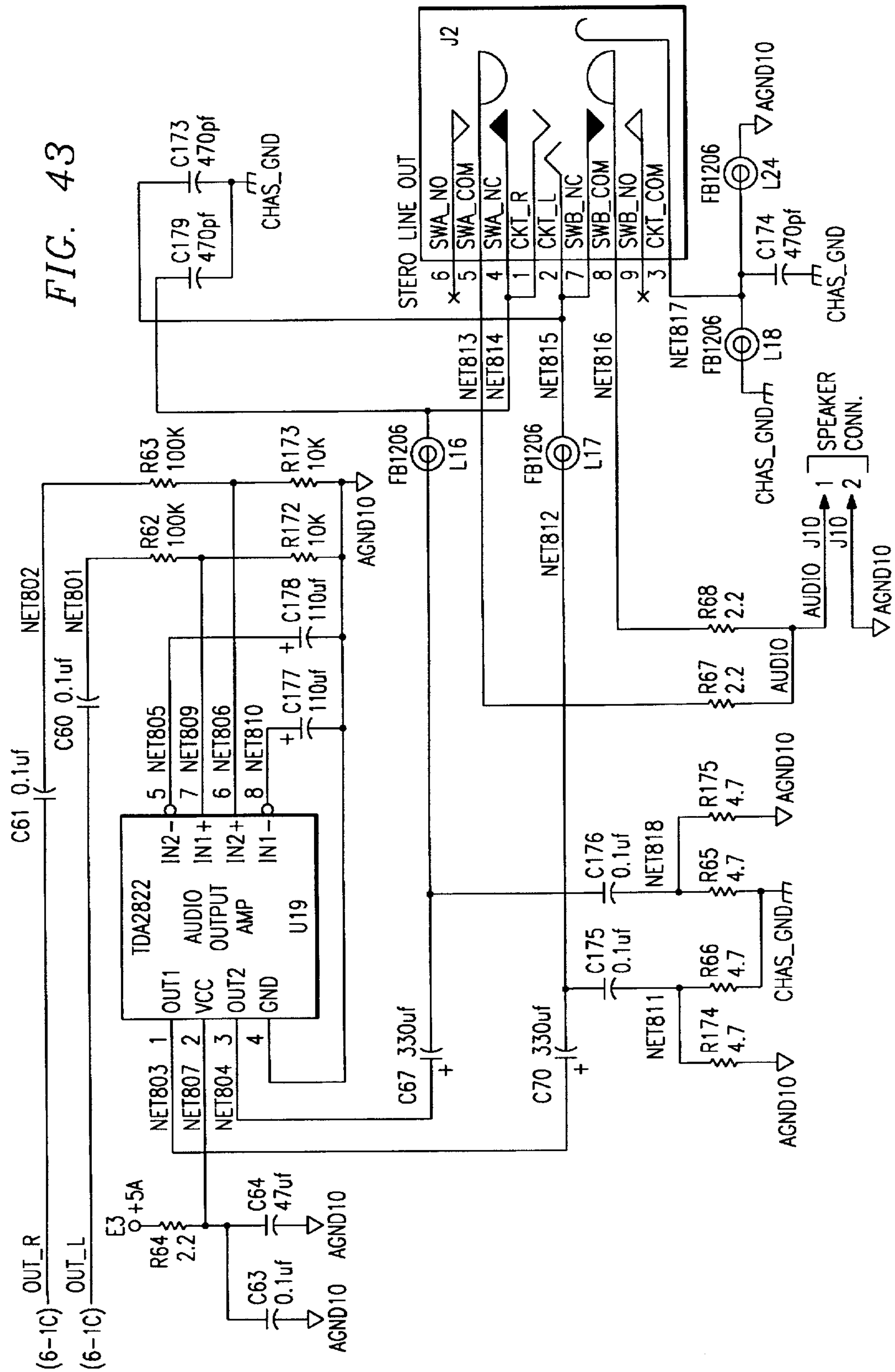

SIOR-
(2-8B)
SIOW-
SYSAEN
J6
44
DSMIC-
(7-8D)
VIDEN-
(12-8D)
SNDRST
(2-4A)
SOUNDEN-
(2-7B)
SNDPWR-
(1-8C)
SNDPWR
(9-6C)
U36
HCT244
SD0
SD1
SD2
SD3
SD4
U6
74HCT14
NET727
NET728
SA8
SA9
SNDWR-
SNDRD-
15ns
16V8F
U27
DECODE ADDRESS 3E4
CLK
OE-
GND
SA0
SA1
SA2
SA3
SA4
SA5
SA6
SA7
SA(0..9)
(2-7D)
HCT273
U37
CK
CL-
RST273-
NET3
(2-2B)
(2-8C)
U17
LF347
+5A
-5A
IN+
IN-
AGND3

FIG. 42B

FIG. 43


J2
STERO_LINE_OUT
6 SWA_NO
5 SWA_COM
4 SWA_NC
1 CKT_R
2 CKT_L
7 SWB_NC
8 SWB_COM
9 SWB_NO
3 CKT_COM
C173 470pf
C179 470pf
CHAS_GND
C174 470pf
FB1206 L24
FB1206 L18
FB1206 L16
FB1206 L17
AGND10
NET813
NET814
NET815
NET816
NET817
NET812
SPEAKER CONN.
J10
AUDIO
R63 100K
R62 100K
R173 10K
R172 10K
C178 110uf
C177 110uf
NET802
NET801
C61 0.1uf
C60 0.1uf
NET805
NET809
NET806
NET810
TDA2822 AUDIO OUTPUT AMP U19
IN2-
IN1+
IN2+
IN1-
OUT1
VCC
OUT2
GND
C67 330uf
C70 330uf
NET803
NET807
NET804
R68 2.2
R67 2.2
R175 4.7
C176 0.1uf
NET818
R65 4.7
C175 0.1uf
R66 4.7
R174 4.7
NET811
E3 +5A
R64 2.2
C64 47uf
C63 0.1uf
OUT_R
OUT_L
(6-1C)
(6-1C)

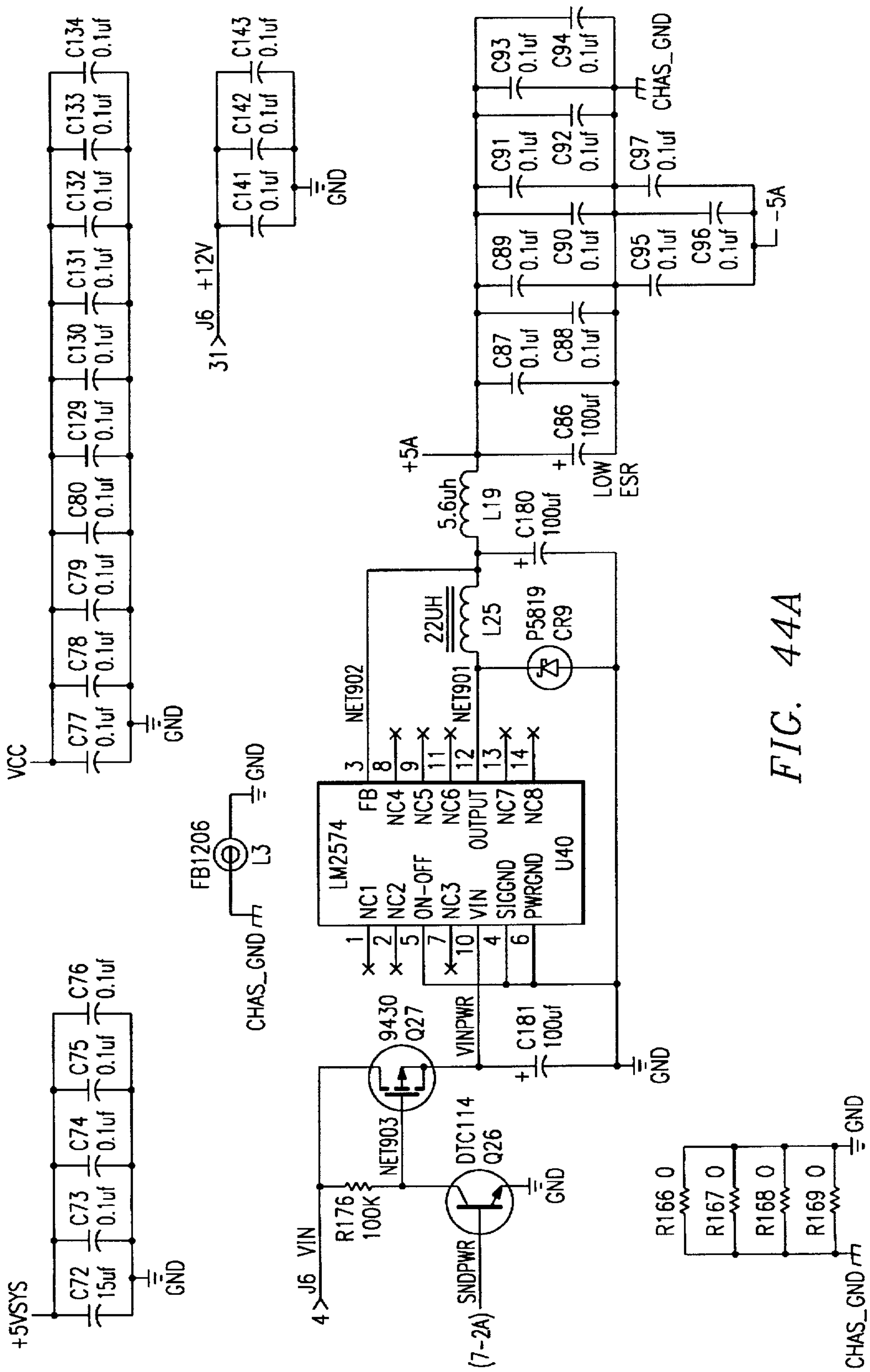

VCC
C77 0.1uf
C78 0.1uf
C79 0.1uf
C80 0.1uf
C129 0.1uf
C130 0.1uf
C131 0.1uf
C132 0.1uf
C133 0.1uf
C134 0.1uf
GND
+5VSYS
C72 15uf
C73 0.1uf
C74 0.1uf
C75 0.1uf
C76 0.1uf
J6 +12V
31
C141 0.1uf
C142 0.1uf
C143 0.1uf
FB1206
L3
CHAS_GND
LM2574
U40
NC1
NC2
ON-OFF
NC3
VIN
SIGGND
PWRGND
FB
NC4
NC5
NC6
OUTPUT
NC7
NC8
NET902
NET901
NET903
22UH
L25
P5819
CR9
5.6uh
L19
C180 100uf
+5A
C86 100uf
LOW ESR
C87 0.1uf
C88 0.1uf
C89 0.1uf
C90 0.1uf
C91 0.1uf
C92 0.1uf
C93 0.1uf
C94 0.1uf
C95 0.1uf
C96 0.1uf
C97 0.1uf
-5A
9430
Q27
VINPWR
C181 100uf
DTC114
Q26
R176 100K
J6 VIN
4
SNDPWR
(7-2A)
R166 0
R167 0
R168 0
R169 0

FIG. 44A

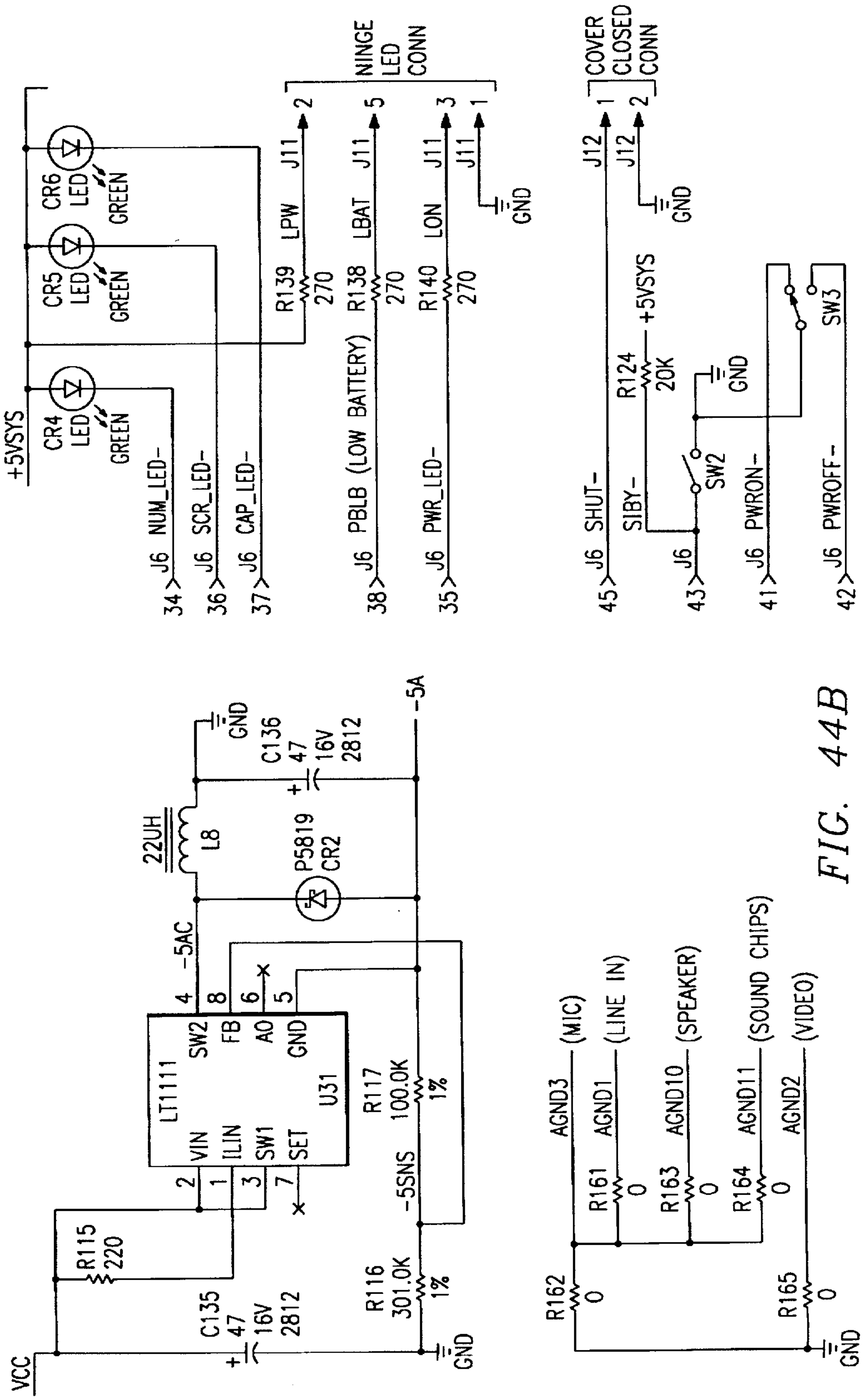

+5VSYS
CR4 LED GREEN
CR5 LED GREEN
CR6 LED GREEN
34 J6 NUM_LED-
36 J6 SCR_LED-
37 J6 CAP_LED-
38 J6 PBLB (LOW BATTERY)
35 J6 PWR_LED-
R139 270
R138 270
R140 270
LPW J11 2
LBAT J11 5
LON J11 3
J11 1
GND
NINGE LED CONN
45 J6 SHUT-
43 J6 SIBY-
41 J6 PWRON-
42 J6 PWROFF-
R124 20K
+5VSYS
SW2
SW3
GND
J12 1
J12 2
COVER CLOSED CONN
VCC
R115 220
C135 47 16V 2812
R116 301.0K 1%
GND
LT1111
U31
VIN
ILIN
SW1
SET
SW2
FB
AO
GND
-5AC
-5SNS
R117 100.0K 1%
22UH
L8
P5819 CR2
C136 47 16V 2812
GND
-5A
R162 0
R161 0
R163 0
R164 0
R165 0
AGND3 (MIC)
AGND1 (LINE IN)
AGND10 (SPEAKER)
AGND11 (SOUND CHIPS)
AGND2 (VIDEO)
GND

FIG. 44B

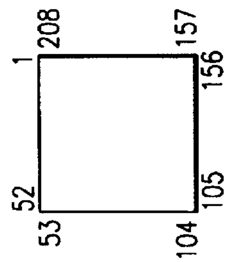
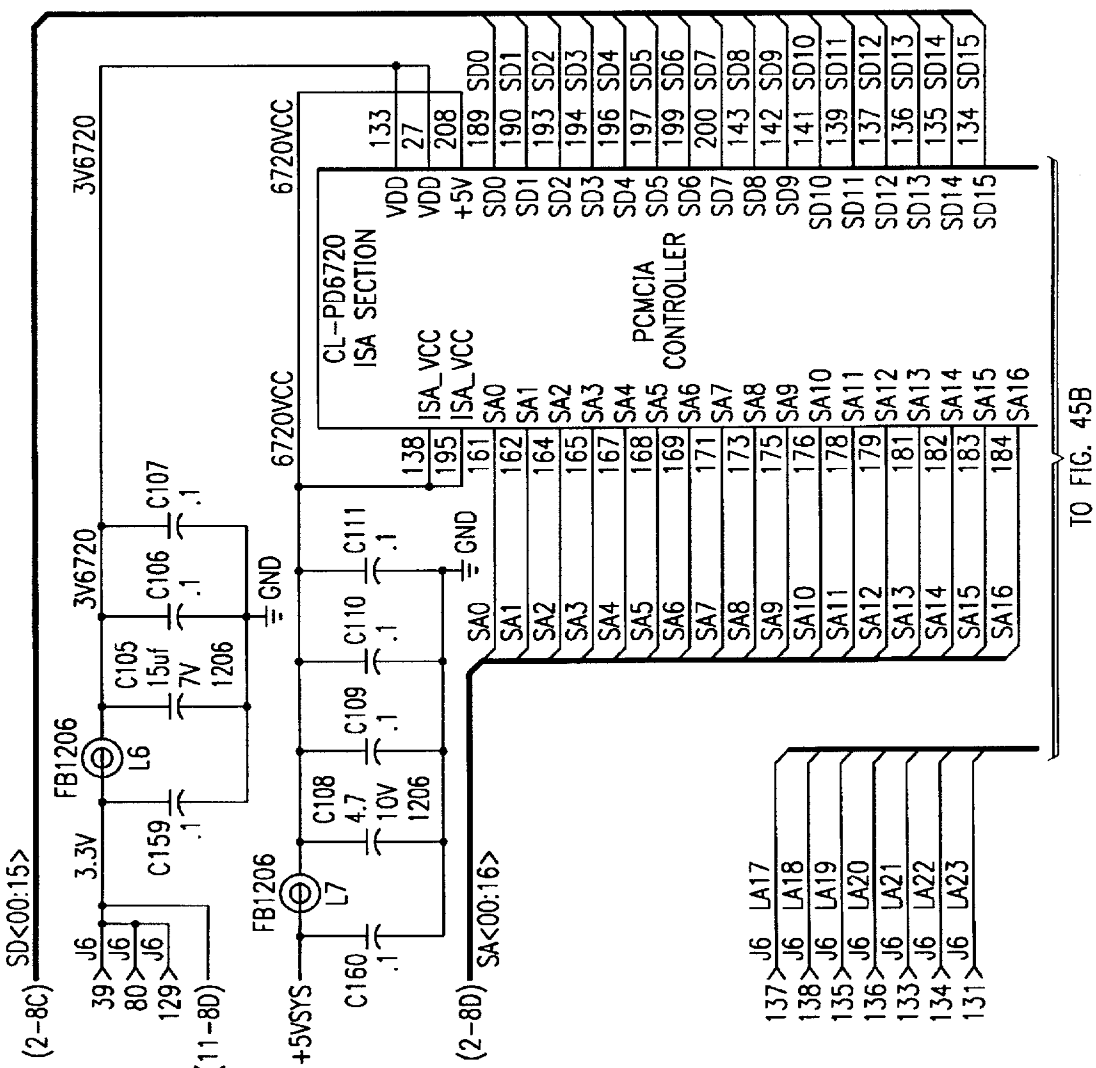

1
208
157
156
52
53
104
105

FIG. 45A


CL–PD6720
ISA SECTION
PCMCIA
CONTROLLER
VDD
VDD
+5V
ISA_VCC
ISA_VCC
6720VCC
6720VCC
3V6720
3V6720
3.3V
+5VSYS
FB1206
L6
FB1206
L7
C105
15uf
7V
1206
C106
C107
C108
4.7
10V
1206
C109
C110
C111
C159
C160
.1
GND
GND
SD<00:15>
SA<00:16>
(2–8C)
(2–8D)
(11–8D)
J6
39
80
129
SD0
SD1
SD2
SD3
SD4
SD5
SD6
SD7
SD8
SD9
SD10
SD11
SD12
SD13
SD14
SD15
SA0
SA1
SA2
SA3
SA4
SA5
SA6
SA7
SA8
SA9
SA10
SA11
SA12
SA13
SA14
SA15
SA16
LA17
LA18
LA19
LA20
LA21
LA22
LA23
137
138
135
136
133
134
131
TO FIG. 45B

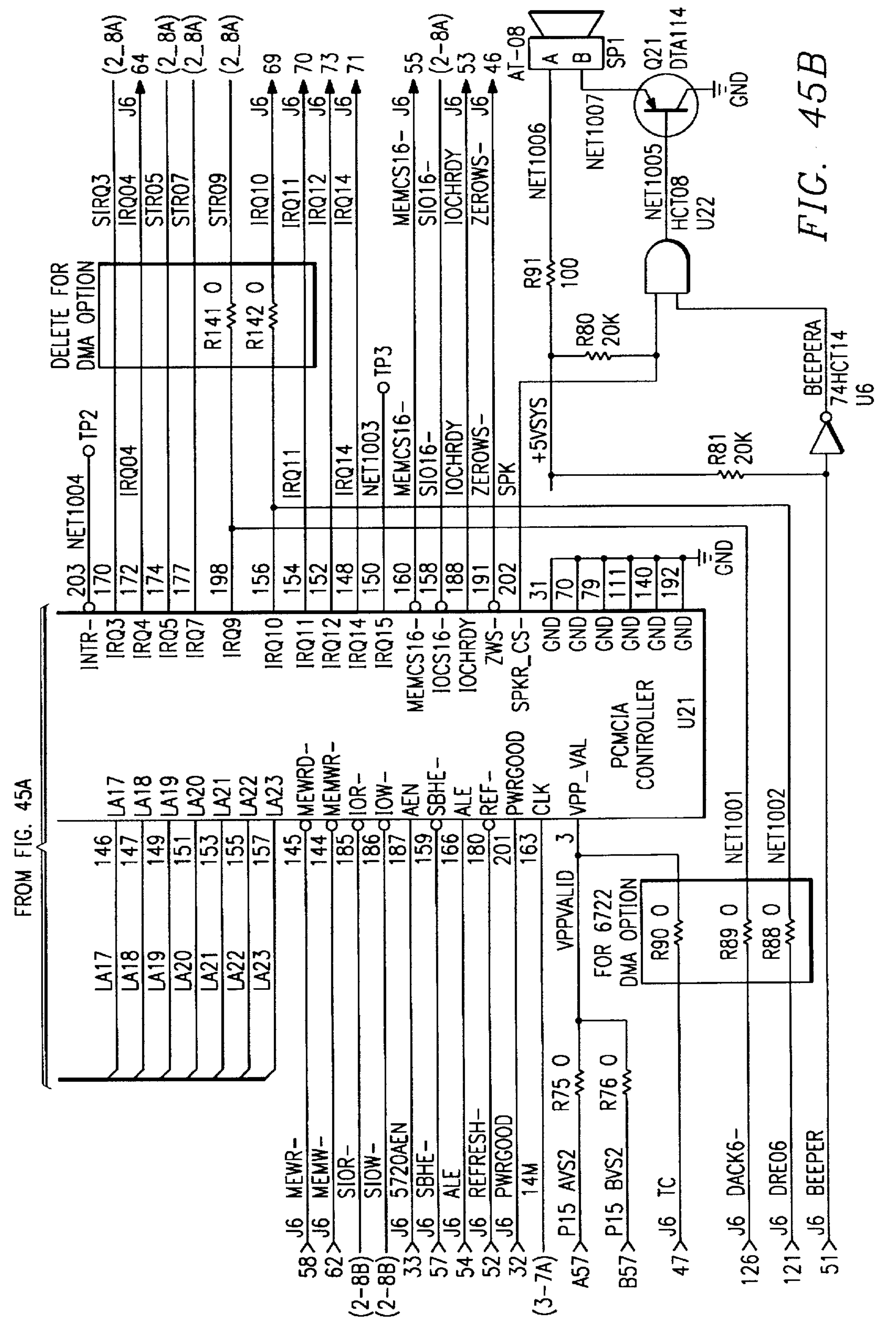

FROM FIG. 45A
PCMCIA
CONTROLLER
U21
DELETE FOR
DMA OPTION
FOR 6722
DMA OPTION
NET1004
TP2
TP3
NET1003
NET1001
NET1002
NET1005
NET1006
NET1007
+5VSYS
R91
100
R80
20K
R81
20K
R141 0
R142 0
R90 0
R89 0
R88 0
R75 0
R76 0
HCT08
U22
Q21
DTA114
AT-08
SP1
BEEPERA
74HCT14
U6
GND
VPPVALID

FIG. 45B

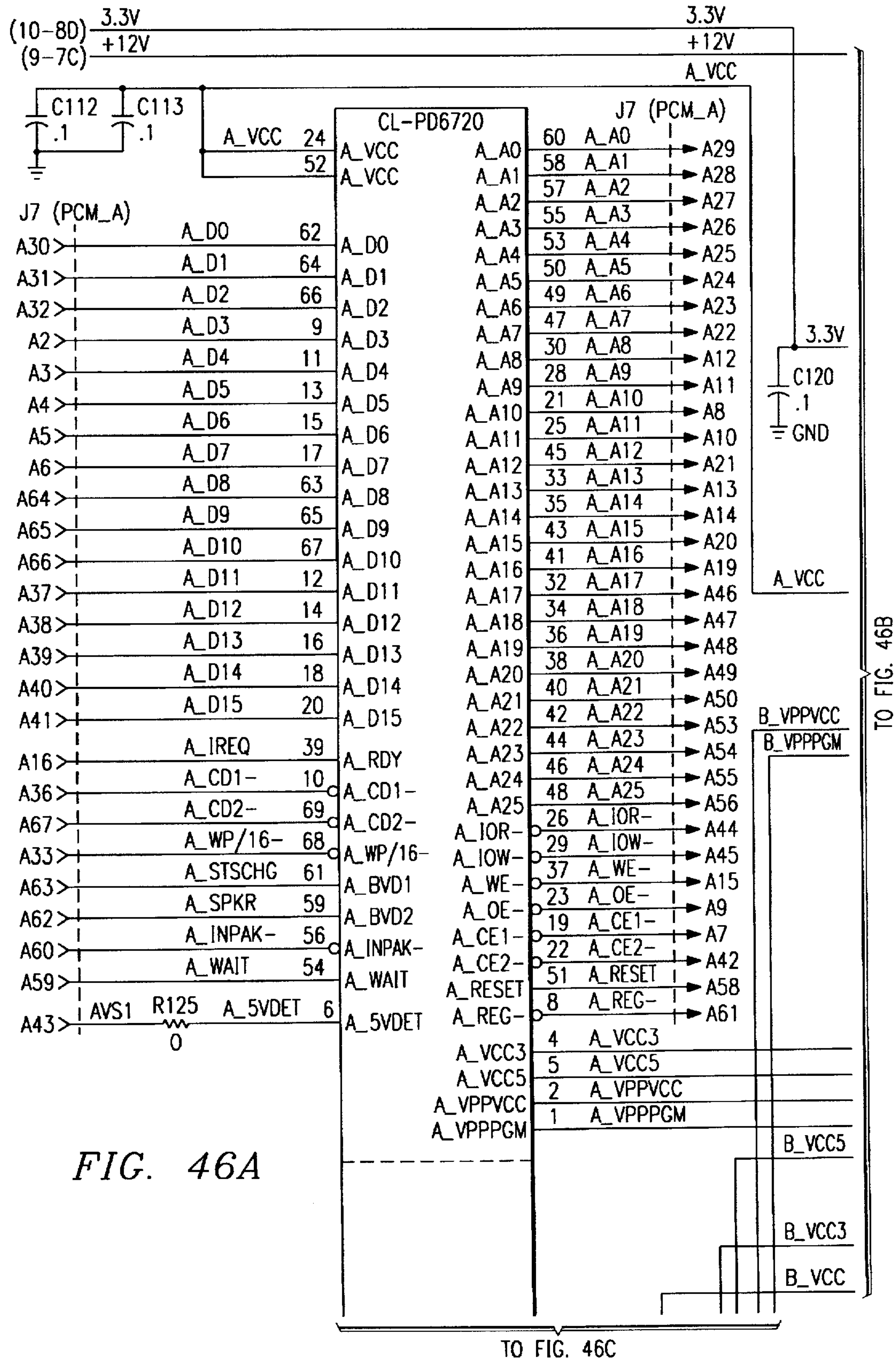
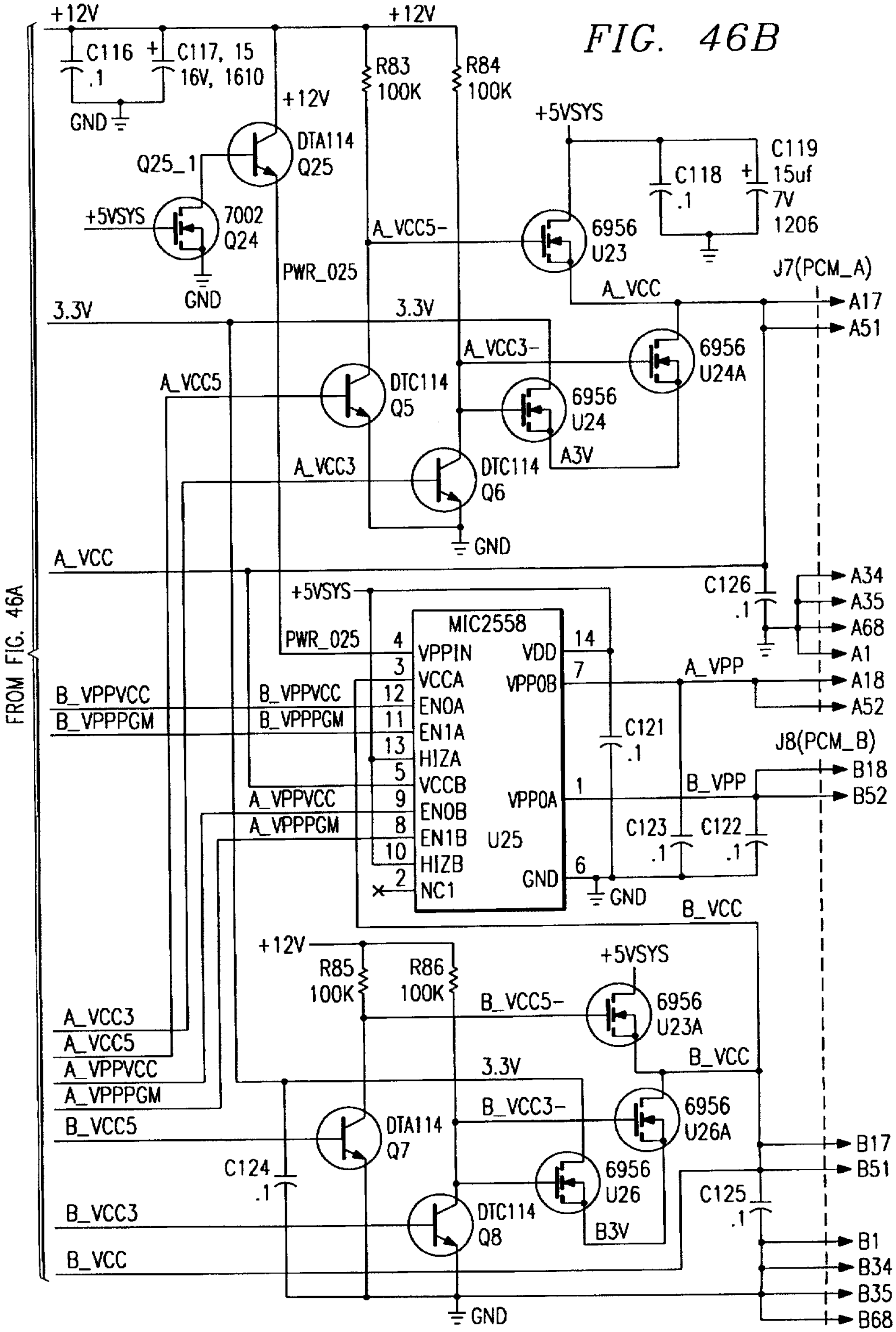

(10-8D) 3.3V
(9-7C) +12V
3.3V
+12V
A_VCC
C112 .1
C113 .1
CL-PD6720
J7 (PCM_A)
J7 (PCM_A)
A_VCC 24 A_VCC
52 A_VCC
A30 A_D0 62 A_D0
A31 A_D1 64 A_D1
A32 A_D2 66 A_D2
A2 A_D3 9 A_D3
A3 A_D4 11 A_D4
A4 A_D5 13 A_D5
A5 A_D6 15 A_D6
A6 A_D7 17 A_D7
A64 A_D8 63 A_D8
A65 A_D9 65 A_D9
A66 A_D10 67 A_D10
A37 A_D11 12 A_D11
A38 A_D12 14 A_D12
A39 A_D13 16 A_D13
A40 A_D14 18 A_D14
A41 A_D15 20 A_D15
A16 A_IREQ 39 A_RDY
A36 A_CD1- 10 A_CD1-
A67 A_CD2- 69 A_CD2-
A33 A_WP/16- 68 A_WP/16-
A63 A_STSCHG 61 A_BVD1
A62 A_SPKR 59 A_BVD2
A60 A_INPAK- 56 A_INPAK-
A59 A_WAIT 54 A_WAIT
A43 AVS1 R125 0 A_5VDET 6 A_5VDET
A_A0 60 A_A0 A29
A_A1 58 A_A1 A28
A_A2 57 A_A2 A27
A_A3 55 A_A3 A26
A_A4 53 A_A4 A25
A_A5 50 A_A5 A24
A_A6 49 A_A6 A23
A_A7 47 A_A7 A22
A_A8 30 A_A8 A12
A_A9 28 A_A9 A11
A_A10 21 A_A10 A8
A_A11 25 A_A11 A10
A_A12 45 A_A12 A21
A_A13 33 A_A13 A13
A_A14 35 A_A14 A14
A_A15 43 A_A15 A20
A_A16 41 A_A16 A19
A_A17 32 A_A17 A46
A_A18 34 A_A18 A47
A_A19 36 A_A19 A48
A_A20 38 A_A20 A49
A_A21 40 A_A21 A50
A_A22 42 A_A22 A53
A_A23 44 A_A23 A54
A_A24 46 A_A24 A55
A_A25 48 A_A25 A56
A_IOR- 26 A_IOR- A44
A_IOW- 29 A_IOW- A45
A_WE- 37 A_WE- A15
A_OE- 23 A_OE- A9
A_CE1- 19 A_CE1- A7
A_CE2- 22 A_CE2- A42
A_RESET 51 A_RESET A58
A_REG- 8 A_REG- A61
A_VCC3 4 A_VCC3
A_VCC5 5 A_VCC5
A_VPPVCC 2 A_VPPVCC
A_VPPPGM 1 A_VPPPGM
3.3V
C120 .1
GND
A_VCC
B_VPPVCC
B_VPPPGM
B_VCC5
B_VCC3
B_VCC
TO FIG. 46B
TO FIG. 46C

FIG. 46A

FIG. 46B


+12V
C116 .1
C117, 15 16V, 1610
GND
+12V
DTA114 Q25
Q25_1
+5VSYS
7002 Q24
GND
R83 100K
R84 100K
+5VSYS
C118 .1
C119 15uf 7V 1206
A_VCC5-
6956 U23
PWR_025
J7(PCM_A)
A_VCC
A17
A51
3.3V
A_VCC3-
6956 U24A
A_VCC5
DTC114 Q5
6956 U24
A3V
A_VCC3
DTC114 Q6
GND
A_VCC
C126 .1
A34
A35
A68
A1
+5VSYS
MIC2558
PWR_025
VPPIN
VDD
VCCA
VPP0B
A_VPP
A18
A52
B_VPPVCC
B_VPPPGM
EN0A
EN1A
HIZA
VCCB
EN0B
EN1B
HIZB
NC1
VPP0A
GND
U25
C121 .1
J8(PCM_B)
B_VPP
B18
B52
A_VPPVCC
A_VPPPGM
C123 .1
C122 .1
GND
B_VCC
FROM FIG. 46A
+12V
R85 100K
R86 100K
+5VSYS
B_VCC5-
6956 U23A
A_VCC3
A_VCC5
A_VPPVCC
A_VPPPGM
B_VCC5
3.3V
B_VCC
B_VCC3-
6956 U26A
DTA114 Q7
C124 .1
6956 U26
B17
B51
C125 .1
B_VCC3
DTC114 Q8
B3V
B_VCC
B1
B34
B35
B68
GND

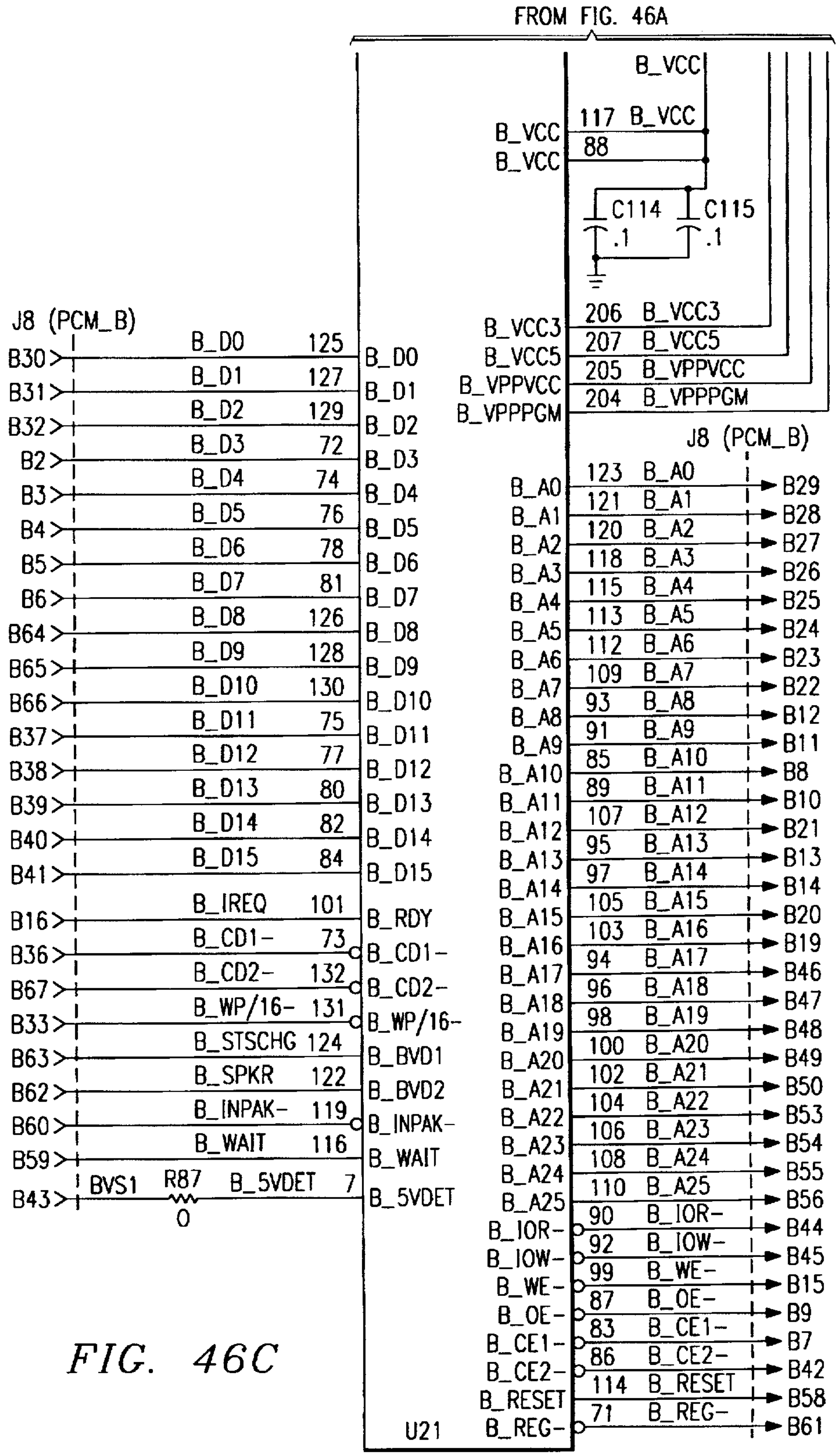

FROM FIG. 46A
B_VCC
117 B_VCC
88
C114 .1
C115 .1
J8 (PCM_B)
206 B_VCC3
207 B_VCC5
205 B_VPPVCC
204 B_VPPPGM
U21
BVS1
R87
0
B_5VDET
B_IREQ
B_RDY
B_CD1-
B_CD2-
B_WP/16-
B_STSCHG
B_BVD1
B_SPKR
B_BVD2
B_INPAK-
B_WAIT
B_IOR-
B_IOW-
B_WE-
B_OE-
B_CE1-
B_CE2-
B_RESET
B_REG-

FIG. 46C

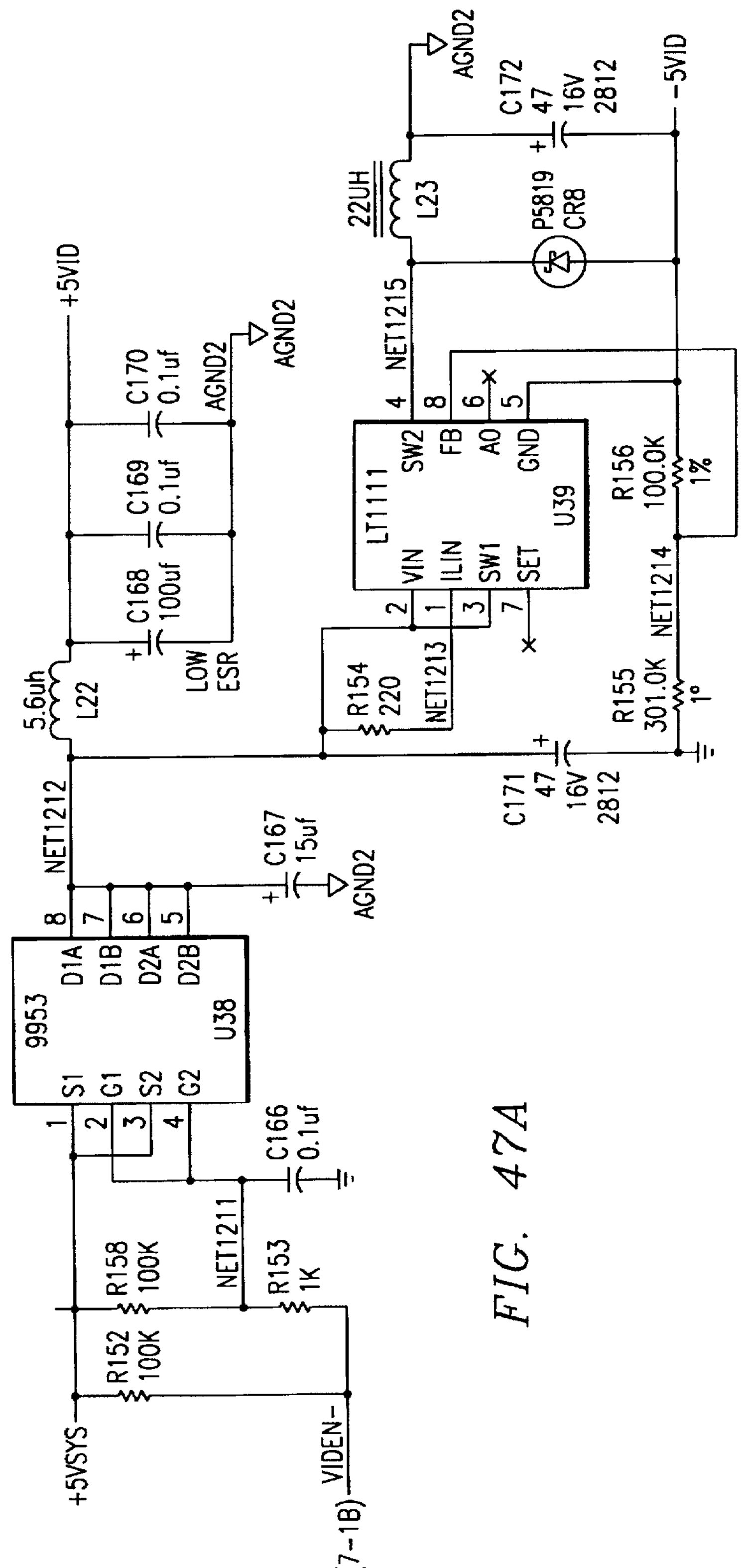
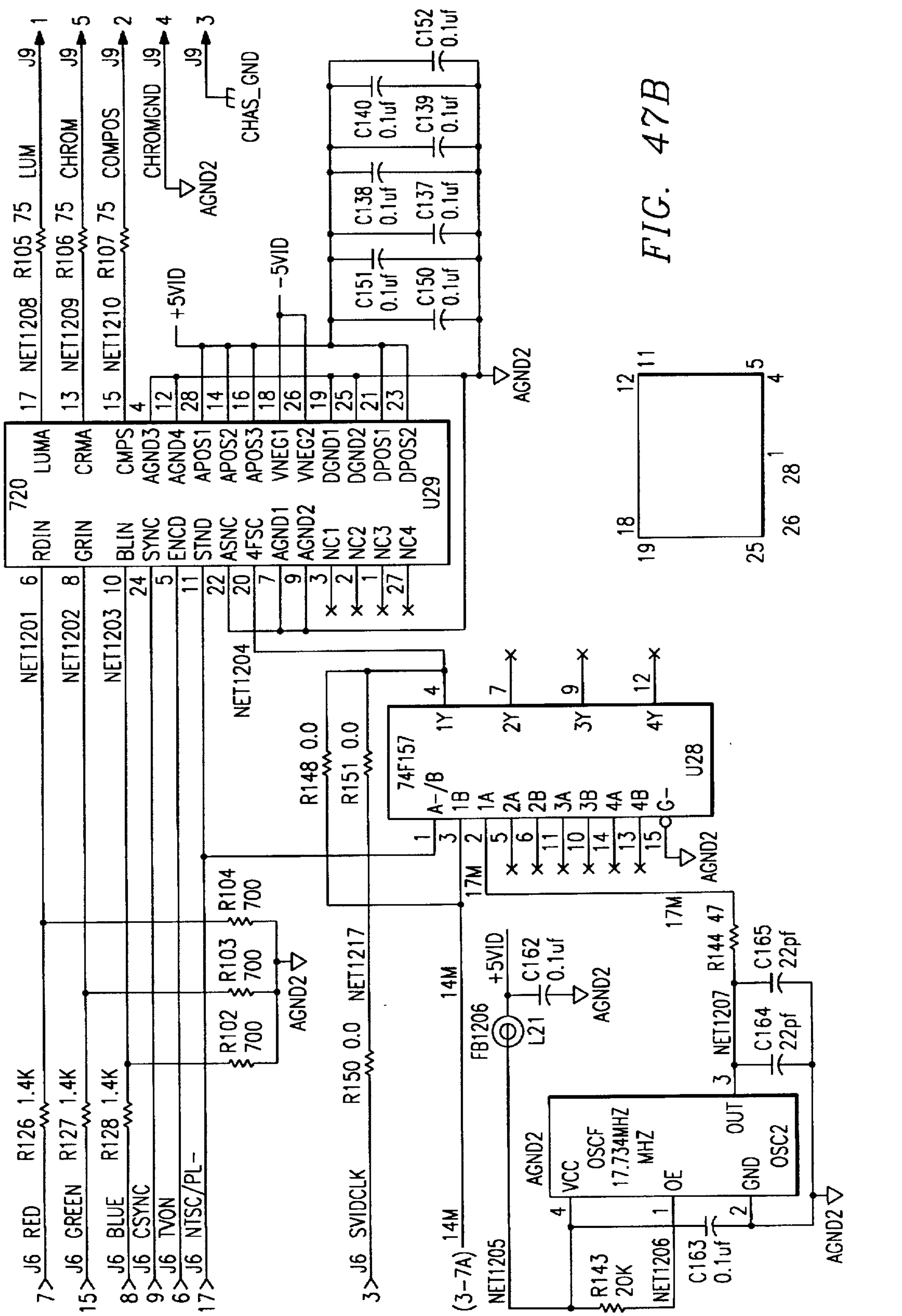

+5VID
AGND2
C170
0.1uf
C169
0.1uf
C168
100uf
LOW
ESR
5.6uh
L22
NET1212
C167
15uf
9953
U38
D1A
D1B
D2A
D2B
S1
G1
S2
G2
C166
0.1uf
R158
100K
NET1211
R153
1K
R152
100K
+5VSYS
VIDEN-
(7-1B)
22UH
L23
P5819
CR8
C172
47
16V
2812
-5VID
NET1215
LT1111
U39
SW2
FB
AO
GND
VIN
ILIN
SW1
SET
R156
100.0K
1%
NET1214
R155
301.0K
R154
220
NET1213
C171
47
16V
2812

FIG. 47A



720
U29
RDIN
GRIN
BLIN
SYNC
ENCD
STND
ASNC
4FSC
AGND1
AGND2
NC1
NC2
NC3
NC4
LUMA
CRMA
CMPS
AGND3
AGND4
APOS1
APOS2
APOS3
VNEG1
VNEG2
DGND1
DGND2
DPOS1
DPOS2
NET1201
NET1202
NET1203
NET1204
NET1208
NET1209
NET1210
R105 75 LUM
R106 75 CHROM
R107 75 COMPOS
CHROMGND
CHAS_GND
AGND2
+5VID
-5VID
C151 0.1uf
C150 0.1uf
C138 0.1uf
C137 0.1uf
C140 0.1uf
C139 0.1uf
C152 0.1uf
J9
74F157
U28
A-/B
1B
1A
2A
2B
3A
3B
4A
4B
G-
1Y
2Y
3Y
4Y
R148 0.0
R151 0.0
R150 0.0
NET1217
R104 700
R103 700
R102 700
R126 1.4K
R127 1.4K
R128 1.4K
J6 RED
J6 GREEN
J6 BLUE
J6 CSYNC
J6 TVON
J6 NTSC/PL-
J6 SVIDCLK
17M
14M
(3-7A)
NET1205
NET1206
NET1207
FB1206
L21
C162 0.1uf
R144 47
C165 22pf
C164 22pf
AGND2
VCC
OSCF
17.734MHZ
MHZ
OE
OUT
GND
OSC2
R143 20K
C163 0.1uf

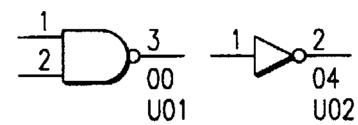
FIG. 47B

NOTES : UNLESS OTHERWISE SPECIFIED :

1. ALL IC DEVICE TYPES ARE PREFIXED WITH SN74.
2. THE FOLLOWING PREFIX'S ARE ALWAY'S USED:
   T IS EQUAL TO "LS"
   AT IS EQUAL TO "ALS"
3. THE FOLLOWING PREFIX'S ARE USED ONLY WHEN INSUFFICIENT CHARACTERS ARE AVAILABLE:
   A IS EQUAL TO "ACT"
   B IS EQUAL TO "BCT"
   V IS EQUAL TO "AS"
   W IS EQUAL TO "AT" OR "ALS"
4. IC PACKAGE TYPE IS INDICATED BY THE FOLLOWING SUFFIX'S:

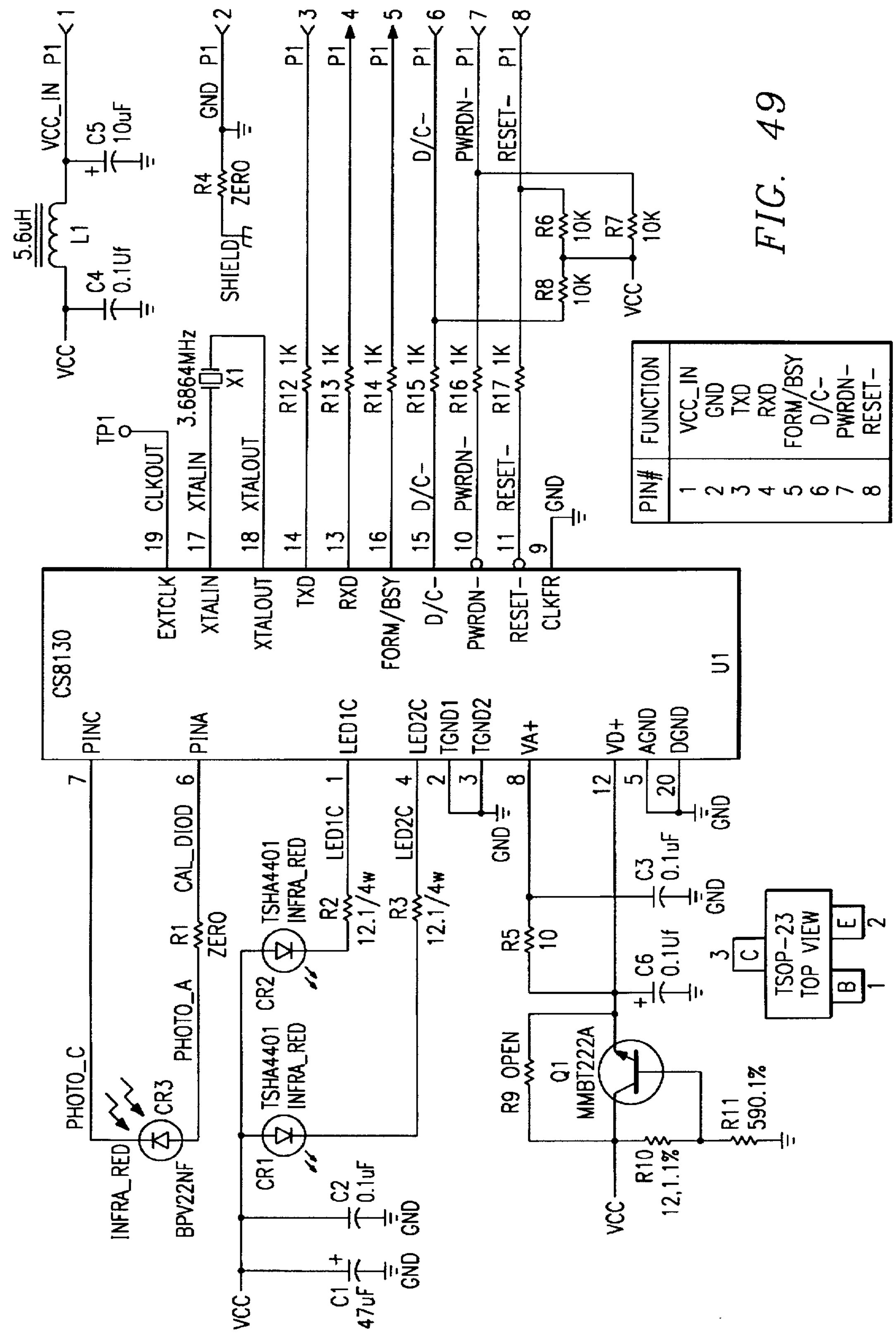
DUAL-IN-LINE, PLASTIC	= "N" OR BLANK
DUAL-IN-LINE, PLASTIC [WIDE]	= NW
DUAL-IN-LINE, CERAMIC	= J
DUAL-IN-LINE, CERAMIC [WIDE]	= JD
CHIP CARRIER, PLASTIC	= F
CHIP CARRIER IN A S.M. SCKT	= FF
CHIP CARRIER IN A PGA SCKT	= FX
CHIP CARRIER, CERAMIC [RECT]	= FE
CHIP CARRIER, CERAMIC [SQUARE]	= FH
FLAT PACKAGE, CERAMIC	= U
FLAT PACKAGE, CERAMIC [WIDE]	= W
GRID ARRAY, PLASTIC	= X
GRID ARRAY, PLASTIC [LIF SCKT]	= XL
GRID ARRAY, PLASTIC [ZIF SCKT]	= XZ
GRID ARRAY, CERAMIC	= Y
GRID ARRAY, CERAMIC [LIF SCKT]	= YL
GRID ARRAY, CERAMIC [ZIF SCKT]	= YZ
SINGLE-IN-LINE	= E,L,M,G
"SOIC", PLASTIC	= D
"SOIC", PLASTIC [WIDE]	= DW
"SOJ", PLASTIC, J LEADS	= R

5. VCC IS APPLIED TO PIN 8 OF ALL 8-PIN IC's, PIN 14 OF ALL 14-PIN IC's, PIN 16 OF ALL 16-PIN IC's, PIN 20 OF ALL 20-PIN IC's, ETC.
6. GROUND IS APPLIED TO PIN 4 OF ALL 8-PIN IC's, PIN 7 OF ALL 14-PIN IC's, PIN 8 OF ALL 16-PIN IC's, PIN 10 OF ALL 20-PIN IC's, ETC.
7. DEVICE TYPE, PIN NUMBERS, AND REFERENCE DESIGNATOR [LOCATION] OF GATES ARE SHOWN AS FOLLOWS:



   00 AND 04 = DEVICE TYPES
   1, 2, AND 3 = PIN NUMBERS
   U01 AND U02 = REF. DESIGNATOR [LOCATION]
8. RESISTANCE VALUES ARE IN OHMS.
9. RESISTORS ARE 1/8 WATT, 5%.
10. CAPACITANCE VALUES ARE IN MICROFARADS.
11. CAPACITORS ARE 50V, 10%.
12. THIS COUPON WILL BE USED ON ALL COMMERICAL MULTILAYER BOARDS.

FIG. 48



PIN#	FUNCTION
1	VCC_IN
2	GND
3	TXD
4	RXD
5	FORM/BSY
6	D/C-
7	PWRDN-
8	RESET-

FIG. 49

ELECTRONIC APPARATUS HAVING A SOFTWARE CONTROLLED POWER SWITCH

CROSS-REFERENCE TO RELATED APPLICATIONS

The following coassigned patent applications are hereby incorporated herein by reference:

Ser. No.	Filing Date	TI Case No.	Title
08/395,335	02/28/95	TI-20391	Real Time Power Conservation and Thermal Management for Computers
08/598,904	12/07/95	TI-20567	Power Management - Thermal

NOTICE

FIELD OF THE INVENTION

This invention generally relates to power switches for electronic devices.

BACKGROUND OF THE INVENTION

Without limiting the scope of the invention, its background is described in connection with desktop and portable computers.

From the advent of electricity, there have been millions of devices built that are powered by electricity. However, every electronic device has to have a method of turning that device on and off. Therefore, virtually every electronic device has a power switch that enables the user to turn that device on and off.

In addition, from the evolution of the computer, there has always been a method and device for turning off a computer's power. In the normal environment, the switch would be turned on to apply power to the computer and turned off to terminate the power. However, the normal power switch simply turns off the power without regard to what the computer is doing at the time. The user simply flips a switch, and thus terminates the power to the computer. Yet, if the computer is in the middle of a software application, or updating a database, or writing to a hard disk, valuable information can be lost or corrupted.

SUMMARY OF THE INVENTION

A need has been discovered for an intelligent power switch; a switch that considers what the computer is doing at the time the user flips the power switch; a switch that will not lose whatever is in the memory at the time; a switch that lets the hard drive position and park its heads before powering down; an intelligent power switch could do this and more.

The present invention solves such a problem. The intelligent power switch can be a mechanical power switch that

is controlled by software. However, the intelligent power switch could also be all electronic and run by software entirely. Or, the intelligent power switch could be a combination of electronics, software and mechanical devices.

The intelligent power switch may be programmed to be intelligent based on what the computer is doing. If the computer is doing something that requires intelligence, (i.e. the system is in a mode that could cause damage to the file system, the communication system, computer network, applications, or even physical hardware damage), then the system knows precisely what to shut down in what order. The software would take control of the power switch away from the hardware and treat that as an event and then process the event at a later time. That would allow preparation for an orderly shut down. The orderly shut down would allow software applications to close files and exit in an orderly manner. In addition, peripheral devices could also shut down orderly. For example, heads on hard drives could be positioned and parked before terminating power. Moreover, peripheral devices connected to the computer serially or by parallel connections could also be shut down in an orderly manner. Further, even display devices could be shut down in an orderly manner.

There are three methods of operating the intelligent power switch One method is to simply terminate the power of the computer whenever the power switch was turned off. Another method is to treat the power switch being turned off as an event and then let the control software proceed with an orderly shut down of the computer's programs and hardware before terminating power to the computer. The last method is similar to the second method, but allows a hardware override after a certain time limit. This would allow the computer to automatically terminate the power in case the software malfunctioned. This hardware override could be implemented as a deadman timer with either a default time limit and/or a time limit that may be adjusted by the control software. In addition, the timer circuit could be setup to allow normal operation if the user quickly turns the power switch to the on position before the system is complete with its orderly shut down. However, the full operation of the system would depend upon how much the system has been shut down already before the user turns on the power again. If, however, the system has not started the shut down procedure, but only registered the event, full operation would begin immediately. Many other variations could also be implemented.

This is a system and method of intelligently terminating power to a computing device. The system may comprise: a processing device; a power source connected to the processing device; a switch connected to the power source; and a control system run by the processing device and connected to the power source and the switch. In addition, the system may include a deadman timer which provides a fail-safe operation. Further, the system may include a means for executing an orderly shut down procedure for software and hardware. Moreover, the system could be tied to a thermal and/or power management system. Additionally, the system could initiate an orderly shut down of peripheral devices connected to the system by serial, parallel or other connections. Other devices, systems and methods are also disclosed.

BRIEF DESCRIPTION OF THE DRAWINGS

In the drawings:

FIG. 1 is a chart of the preferred embodiment;

FIGS. 2 is a flow chart of the bootup process of a computer;

FIG. **3** is a circuit diagram of an embodiment of hardware used for the Intelligent power switch;

FIG. **4** is a block diagram of the electronic architecture of a basic computer;

FIG. **5** is an isometric view of a portable computer;

FIG. **6** is a block diagram of the portable computer of FIG. **4**;

FIG. **7** is an exploded view of a portable computer;

FIG. **8** is a closeup of the main printed circuit board from FIG. **7**;

FIGS. **9–30D** show logic diagrams of an implementation of the main printed circuit board of FIG. **7**;

FIGS. **31–35** show logic diagrams of an implementation of the keyscan printed circuit board of FIG. **7**;

FIGS. **36–47B** show logic diagrams of an implementation of the PCMCIA/Sound printed circuit board of FIG. **7**; and

FIGS. **48–49** show logic diagrams of an implementation of the IR module printed circuit board of FIG. **7**.

Corresponding numerals and symbols in the different figures refer to corresponding parts unless otherwise indicated.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

The intelligent power switch can be executed in any combination of three methods. The first is to have the intelligent power switch execute in simple mode; when the user turns off the power, the power switch simply terminates the power to the computer. The second method makes the power switch intelligent by controlling it with software; this method will not turn the power off until the software program releases control and triggers the termination of power. The third method is to have a deadman timer run concurrently to the software program and time out after a specified time limit and then proceed to terminate the power to the computer.

A software program that controls the power switch can be executed on a central processing unit (CPU), or a separate processor like an application processor. For example, you can either have a CPU run the control program with its other programs, or dedicate a small microprocessor to monitor the power switch.

The software control program can have three modes of operation: no intelligent power switch, intelligent power switch with real time event, or intelligent power switch with delayed event (for the sake of clarity, real time events and delayed events will be described with real time events getting attention by the CPU and other hardware in real time, similar to interrupts, and delayed events getting attention from the CPU at a later time like any other software program getting scheduled time slices). The software control program allows the applications time to shut down in an orderly manner. However, if any of the application programs lose control or have some type of unrecoverable error, and are not able to get back to the timer before it runs out, the deadman timer will time out and allow the computer to shut down as in the simple mode, just like an ordinary power switch. Therefore, the setting of the deadman timer is crucial; the time limit should be long enough to let the applications shut down in orderly process, and get back to the timer and reset it if necessary. In addition, the time limit should not be too long, in case the applications get into some type of unrecoverable error; the user should not have to wait too long for the computer to shut off.

FIG. **1** describes a general flow of the intelligent power switch. The system begins by starting the software control program **10**. The software control program may be started at the bootup process or on user demand. However, after the software control program starts, the timer circuit gets set **12**. The timer circuit may get set to a value by the software, or have a default value. However, the timer must be able to get reset by the software control program. Once the timer circuit gets set, the timer proceeds until timed out **14**. In addition, the software program initiates an orderly shut down procedure **18** concurrently. The software program could first start the software shut down process **20** and the hardware shut down process **22**. However, these two procedures could be implemented in any order or intermixed. Yet, the software program has to be able to reset the timer circuit **16** before it times out if additional time is needed to complete the shut down process. Finally, after the timer circuit has timed out, the orderly termination of power to the system begins **24**. In addition, the software program could be implemented to set the timer value to time out instantly if the shut down process is complete.

FIG. **1** details a flowchart of the software control program. However, as stated before, the software control program can be implemented at different times in the operation of the control program. It may be implemented in the bootup process of the computer and have the user turn it off or on. It may also be implemented only when the user hits the power switch to turn the computer off. It could be implemented with a battery source to supply just enough power to ensure an orderly shut down process of the peripherals and the application programs. This implementation would be beneficial in case of a power failure. In addition, the user could turn on the software control program at any time by just executing the program.

FIG. **2** details the implementation of the software control program in the preferred embodiment. First, the basic input/output system (BIOS) is initialize **26**. Then, the operating system gets initialized **28**. Within the operating system initialization, some of the steps executed are: initialize core operating system, initialize advanced power management system, start scheduler, and start user interface. Once the operating system is initialized, the other software applications may be implemented **30**. Even though the software control program may be implemented in other stages of the system's operation, the software control program is implemented at this point in the preferred embodiment.

An example implementation of the software control program is included in this specification. However, the invention could be implemented in a multitude of ways and is not restricted to this embodiment.

DEADMAN TIMER CIRCUIT OPTION

Deadman Timer

An optional feature of the present invention can ensure a shut down of the device even though the computer software has malfunctioned. This optional feature is a fail-safe or a deadman timer circuit built into the intelligent power switch. The deadman timer would function after the intelligent power switch loses software control After a specified period of time elapses that would indicate the software has lost control, then the switch will revert back to turning power off in an unconditioned state just as if it had not been an intelligent power switch. This deadman timer is a fail-safe condition.

However, with the creation of a software controlled power switch, sometimes the software monitors the intelligence malfunctions. In addition, the software may malfunction because of the processor that it's running on.

Once the software has control of the power switch, the hardware circuit sets a maximum time that the circuit will wait for a response from the software control. If the circuit does not get a response from the software control, it will shut down the rest of the system. However, the software control can come back to the circuit and reset the clock, or even set a new maximum time for the circuit to wait for another response. This would enable the software control program to be more dynamic in case it needs to wait for unexpected events before powering the system down. This mode would place a burden on the software to come back to reset the timer every so often before the expiration of the maximum time. However, the user may also adjust the maximum time. This versatility would allow the user to determine what is acceptable as the maximum time for circuit to wait.

When the power switch is turned on, the system boots; the software boots; the Basic Input/Output System (BIOS) initializes and then the dead man timer gets set to zero and the power switch gets turned to simple, the default condition. Then, through the process of initializing the rest of the computer system (the software, and the different sets of hardware), the software control program will determine whether to turn on the intelligence power switch. However, the software control program may also determine to wait until the operating system is running or to wait until the user selects a particular application to turn this switch on or off.

The software control program is a real time event. It may be turned on or off based on the boot up condition. Then the software control program can decide whether to continue to keep it on or keep it off or whether to come back later and turn it to intelligent or simple mode.

For example, in the preferred embodiment, the computer can go through the boot process, then load DOS or Windows and then turn the intelligent power switch on. The software control program has to get back at least every 15 seconds or the system is going to turn itself off because the deadman timer switch is on.

In addition, the power switch can be programmed to watch a System Management Interrupt (SMI). The SMI can either be acted on real time, or can be acted upon later.

If the power switch is set to be acted upon a real time event, then as soon as the event is triggered, the heads of the hard drive are positioned and parked. Then the power from the hard drive is turned off, along with the power to the displays and other devices within the system. Then the CMOS parameters that need to be saved are saved. This process would enable protection against lost clusters or allocations on hard disks, which is a major problem on other products.

If the power switch is set to be acted upon a delayed event, then the software control program allows the operating system and other programs to prepare for shut down. This will allow the system to go through and start closing files. In addition, it will start updating any pertinent parameters and then trigger the event to start the shut down process. The shut down process is the same as the previous scenario.

In both scenarios, the power switch may be tied directly to the actions required by the operating system to do an orderly shut down. However, the intelligent power switch can also be integrated into an existing shut down software program (e.g. Super Shutdown by Texas Instruments Incorporated). This would allow the shut down program to automatically go through all the software programs and ensure an orderly shut down. A shut down program could ensure that all files are closed, and parameters updated before it gives control back to the software control program.

DEADMAN TIMER CIRCUIT IMPLEMENTATION

The intelligent power switch circuit with the deadman timer consists of the five functional parts identified in FIG. **3** and as described below:

Supervisory Transistor **38**—Low power transistor switch that runs unregulated input power on and off to the computer power supervisory circuitry.

Manual Switch **56**—Manual power on/off switch that is set by the computer operator and that informs the Intelligent power switch and computer processor to turn system power on and off.

Power Off Timer **74**—provides power control to processor when Intelligent power switch is in the intelligent mode and Manual Switch **56** is in the "off" position.

Power Off Latch **36**—system power off latch holds computer power "off" when processor has turned off power and Manual Switch **56** is still "on".

Power Off Latch Trigger **32**—provides power control by processor when Intelligent power switch is in the intelligent mode and the Manual Switch **56** is in the "on" position.

The Intelligent power switch control signals shown in FIG. **3** are described below:

"VIN"—unregulated DC input power to computer sourced by external power and/or internal batteries.

"VINS"—unregulated DC input power to computer power Supervisory Circuit **38**.

"NVCC"—regulated DC power to computer logic.

"REF 2.5"—compalitor reference voltage.

"SFTOFF"—low active logic signal from processor indicating software status of the manual on/off switch.

"PWROFF"—low active logic signal from manual on/off switch indicating "off" position.

"PWRON"—low active logic signal from manual on/off switch indicating "on" position.

"PWRSWON"—logic signal to processor indicating status of the manual on/off switch.

"SMPL"—logic signal from processor indicating mode of Intelligent power switch.

"TRMRRST"—low active logic signal from processor that resets the Power Off Timer **74**.

The Intelligent power switch Circuit shown in FIG. **3** couples the computer operator and the computer processor to the computer system power switch. The computer processor can be programmed to turn off the system power intelligently. The computer power is turned on by the operator changing the manual switch status from "off to "on". In addition, the computer power may be turned off by the computer processor under software controlled conditions in an orderly and intelligent manner, through the intelligent power switch circuit. The manual power on/off switch **56** may be a single pole, double throw as shown in FIG. **3**, or may be any switching device that provides compatible logic levels when connected to the circuit. The Intelligent power switch defaults to the simple mode when system power is "off" or when the system power "on" routine is being executed by the computer logic (the computer boot-up process). The intelligent switch can be changed to the intelligent mode by the computer at any time after the power "on" routine is complete or as part of the system initialization during the power "on" routine. The Intelligent power switch is in the simple mode whenever the logic signal "SMPL" is low. In this mode, the system power can only be

turned "on" and "off" by the operator using the manual switch **56**. The power "off" timer **74** and power "off" latch trigger **32** are disabled through diode **62** and transistor **60** through diode **34** respectively, when signal "SMPL" is held low by the computer or by loss of system power. Manual power "on" and "off" in this mode is as follows:

1) The closed contacts of the manual switch **56** in the "off" position grounds signal "PWROFF" and disables the power off latch trigger **32** through diodes **46** and **40**, and clear the power off latch **36** through diode **46**. In addition, the open contact of the manual switch **56** in the "off" position allows the signal "PWRON" to float up through resistors R7 and R6, shutting off the supervisory transistor **20**. The system is turned off in this mode.
2) The closed contacts of the manual switch **56** in the "on" position grounds signal "PWRON" and turns on the supervisory transistor **38** through resistor **64**. The system power is turned on in this mode. In addition, the open contact of the manual switch **56** in the "on" position turns off diodes **46** and **52** enabling the power off latch **36** and signaling to the processor that the manual switch **56** is "on" by pulling up signal "PWRSWON" through resistor **20**.

The intelligent mode of the Intelligent power switch can only be set by the computer processor when the system power is on. The processor enables the intelligent mode by setting the signal "SMPL-" high. This enables the power off latch trigger **32** circuit by turning off diode **34** and enables the power off timer **74** by turning off diode **62** and turning on transistor **60**. Transistor **58** is turned on by signal "SFTOFF" being high, and the power off timer **74** is held reset by the low signal "TMRRST-". The output of the power off latch trigger **32** is held low by the high level on the inverting input of 32 from the signal "SFTOFF". Thus the power off latch **36** is not triggered and system power remains on.

System power can now be turned off with the Intelligent power switch in the intelligent mode and the manual power switch **56** in the "on" position in the following way only:

The computer processor sets the signal "SFTOFF" low, turning off the latch trigger comparitor output **32**. This sets the power off latch **36** by turning on transistor **44** through diode **40** and resistor **22** to the +5 VDC on signal "NVCC". Transistor **44** turns on transistor **24** which holds transistor **44** on. Transistor **24** also turns on diode **36** which turns off transistor **38**, thus turning off system power. The power off latch **36** remains set as long as the manual power switch **56** remains in the "on" position and power lasts on signal "VIN" from the external and/or internal unregulated power sources.

The computer processor can control the system power with the intelligent switch in the "Intelligent" mode and the manual switch in the "off" position. The supervisor transistor **20** is held on through resistor **54** so long as transistors **58** and **60** and the time-out comparitor **74** are all turned on. The system power is turned off if any one of the three are turned off.

The computer processor can now turn off system power by setting signal "SFTOFF" low, turning off transistor **60**, or by allowing the power off timer **74** to turn off when capacitor **72** charges through resistor **68** to a level above the voltage at the junction of the divider at resistors **64** and **66**, or by allowing the power off timer **74** to turn off after a software controlled time that holds signal "TMRRST" low.

Giving power off control to the computer processor insures that the shut down is done in an orderly and predictable manner protecting function integrity for the user.

Listed below in Table 1 are examples of types of devices and values that can be implemented in the Intelligent power switch circuit illustrated in FIG. **3**. It is to be understood that the present invention is not limited to only this embodiment.

TABLE 1

Element	Name	Description
20	Resistor	10k resistor
22	Resistor	47k resistor
24	Transistor	DTA transistor
26	Resistor	47k resistor
28	Capacitor	.0047 f capacitor
30	Resistor	4.7k resistor
32	Invertor	TLC393C/2 invertor
34	Diode	BAT54A diode
36	Diode	BAT64 diode
38	Transistor	2907 transistor
40	Diode	BAT54A diode
42	Resistor	10k resistor
44	Transistor	DTC transistor
46	Diode	BAV70 diode
48	Capacitor	0.1 f capacitor
50	Resistor	47k resistor
52	Diode	BAV70 diode
54	Resistor	4.7k resistor
56	Switch	single pole, double throw
58	Transistor	BST82 transistor
60	Transistor	BST82 transistor
62	Diode	BAT54A diode
64	Resistor	10k resistor
66	Resistor	3.3k resistor
68	Resistor	1 M resistor
70	Diode	BAT54A diode
72	Capacitor	10/16 v capacitor
74	Invertor	TLC393C/2 invertor

OTHER OPTIONS

The control software program can also be interactive. It can prompt the user with questions like "do you really want to turn the power off—yes or no?" and if the user says yes, then the program could go ahead and execute an orderly shut down. However, the program could also just tell the user how to manually execute an orderly shut down, and let the user manually shut down the software programs and/or hardware. For example, the program could have the user close all files, close all software programs in a specific order, and then turn off all hardware devices that are hooked up to the computer. Yet, the control program could also be set to execute an orderly shut down automatically. In addition, the interactive part could more or less interactive, depending on options set in installation, in execution or at production.

Another option that could be implemented, is to automatically shut down the computer when it goes into an uncontrollable state. This could be done if the deadman timer was set to a time that the software control program knew it could get back to the timer if the computer was in a controllable state. However, if a software program took control of the computer and then went into an infinite loop or some other uncontrollable state, the deadman timer would time out and then execute the shut down procedure. Again, the intelligent power switch could be set to just simply terminate the power to the computer or go through an orderly shut down first. Moreover, the deadman timer could also run in the background while the control program is executing an orderly shut down, and then time out if the software control program gets in an uncontrollable state.

The software control program may also interface with a thermal management system (i.e. the thermal management systems described in U.S. patent application Ser. No.

08/395,335 and U.S. patent application Ser. No. 08/568,904) and/or a power management system (i.e. the power management system described in U.S. patent application Ser. No. 08/395,335). This would allow an intelligent power switch to have an orderly shut down when the user turns the computer off, and would also allow the features of the thermal and power management systems to be integrated into the intelligent power switch. The thermal and/or power management systems could control the deadman timer and reset to zero when the system wanted to terminate power. This would be helpful if the computer was in imminent danger of overheating, or in some other state of impending danger.

If the intelligent power switch incorporates the power management system, the software control program can be tied off of Advanced Power Management™ (APM) events under Windows 3.11™ and Windows95™ (Advanced Power Management, Windows 3.11 and Windows95 are trademarks of Microsoft). This would allow the software control program to be posted to the 530B interrupt. This would ensure that the operating system will check once every one to five seconds to make sure the software control program is still alive. Other operating systems might implement other interrupts that the software control program could be linked to also. In addition, the 530B interrupt may also change in implementation in other versions of the Windows™ operating system. However, the software control program would still function as long as it was checked periodically. For further details on the implementation of the preferred embodiment, refer to the APMfuncb procedure, as well as the SMI interrupt procedure in the example software implementation included at the end of the specification.

In sum, the present invention can be a mechanical power switch that is controlled by software. However, the intelligent power switch could also be all electronic and run by software entirely. Or, the intelligent power switch could be a combination of electronics, software and mechanical devices.

In addition, the intelligent power switch may be programmed to be intelligent based on what the user is doing. If the user is doing something that requires more intelligence, (i.e. the system is in a mode that could case damage to the file system, the communication system, computer network, applications, or even physical hardware damage), then the system knows precisely what to shut down in what order. The software would take control of the power switch away from the hardware and treat that as an event and then process the event at a later time. That would allow preparation for an orderly shut down. The orderly shut down would allow software applications to close files and exit in an orderly manner. In addition, peripheral devices could also shut down orderly. For example, heads on hard drives could be positioned and parked before terminating power. Moreover, peripheral devices connected to the computer serially or by parallel connections could also be shut down in an orderly manner. Further, even display devices could be shut down in an orderly manner.

There are three methods of operating the intelligent power switch. One method is to simply terminate the power of the computer whenever the power switch was turned off. Another method is to treat the power switch being turned off as an event and then let the control software proceed with an orderly shut down of the computer's programs and hardware before terminating power to the computer. The last method is similar to the second method, but allows a hardware override after a certain time limit. This would allow the computer to automatically terminate the power in case the software malfunctioned. This hardware override could be implemented as a deadman timer with either a default time limit and/or an time limit that may be adjusted by the control software. In addition, the timer circuit could be setup to allow normal operation if the user quickly turns the power switch to the on position before the system is complete with its orderly shut down. However, the full operation of the system would depend upon how much the system has been shut down already before the user turns on the power again. If, however, the system has not started the shut down procedure, but only registered the event, full operation would begin immediately. Many other variations could also be implemented.

FIGS. **4**–**8** depict example devices that the present invention can be implemented on. However, these embodiments are not intended to be limiting. The present invention may also be implemented on other devices as well.

While some of the embodiments shown are in relation to a portable computer, the present invention can also be integrated into any electronic device. For example, the present invention could be implemented on a main frame, mini, desktop, or portable computer. FIG. **6** is a block diagram of a basic computer **900** upon which the present invention could be implemented. Computer **900** comprises a Power Input and Conversion Unit **905** having power input **910**. Unit **905** senses the input conditions and selects appropriate circuitry to convert the input to the voltages needed to power the other elements of the system. Output from the conversion unit is coupled to Bus **915**, which comprises paths for power as well as for digital information such as data and addresses.

Bus **915** typically needs more than one power line. For example, the motor drive for a hard disk requires a different power (voltage and current) than does a CPU, for example, so there are parallel power lines of differing size and voltage level in Bus **915**. A typical Bus **915** will have, for example, a line for 24 VDC, another for 12 VDC, and yet another for 5 VDC, as well as multiple ground lines.

Bus **915** connects to a video display controller **920** including Video Random Access Memory (VRAM) which both powers and controls display **925**, which in a preferred embodiment is a display driven by analog driver lines on an analog bus **930**. Bus **915** also connects to a keyboard controller **935** which powers and controls keyboard **940** over link **945**, accepting keystroke input and converting the input to digital data for transmission on Bus **915**. The keyboard controller may be physically mounted in the keyboard or within the computer housing.

Bus **915** comprises, as stated above, both power and data paths. The digital lines are capable of carrying 32 addresses and conveying data in 32 bit word length. To minimize pin count and routing complexity, addresses and data are multiplexed on a single set of 32 traces in the overall bus structure. One with skill in the art will recognize that this type of bus is what is know in the art as a low-pin-count or compressed bus. In this kind of bus different types of signals, such as address and data signals, share signal paths through multiplexing. For example, the same set of data lines are used to carry both 32-bit addresses and data words of 32-bit length.

In Bus **915**, some control signals, such as interrupt arbitration signals, may also share the data lines. Typical examples of buses that are exemplary as usable for Bus **215** (with the exception of power supply analog lines in Bus **915**) are the IIS-Bus" implemented by Sun Microsystems, the "Turbochannel" Bus from Digital Equipment Corporation,

and buses compatible with the IEEE-488 standard. Bus **915** is also a high-speed backplane bus for interconnecting processor, memory and peripheral device modules.

CPU **950** and RAM **955** are coupled to Bus **915** through state translator **960**. CPU **950** may be of a wide variety of CPUs (also called in some cases MPUS) available in the art, for example Intel 80386 or 80486 models, MIPS, RISC implementations, and many others. CPU **950** communicates with State Translator **960** over paths **965**. State Translator **960** is a chip or chip set designed to translate commands and requests of the CPU to commands and requests compatible with Bus **915**. It was mention previously that CPU **950** may be one of a wide variety of CPUs, and that Bus **915** may be any one of a wide variety of compressed busses. It will be apparent to one with skill in the art that there may be an even wider variety of state translators **960** for translating between the CPU and Bus **915**.

RAM memory module **955** comprises conventional RAM chips mounted on a PCB as is known in the art, and connectable to state translator **960**. Preferably, the RAM module is "on board" the CPU module to provide for rapid memory access, which will be much slower if the RAM is made "off board". As is the case with Bus **915**, paths **965** and **970** comprise power and ground lines for CPU **950** and Translator **960**.

FIG. **5** illustrates a portable personal computer **800** having a display **810** and a keyboard **820**. The present invention is ideally suited for the portable computer **800**.

FIG. **6** is a block diagram of portable computer **800**. Portable computer **800** is a color portable notebook computer based upon the Intel Pentium microprocessor. Operating speed of the Pentium is 75 Mhz internal to the processor but with a 50 Mhz external bus speed. A 50 Mhz oscillator is supplied to the ACC Microelectronics 2056 core logic chip which in turn uses this to supply the microprocessor. This 50 Mhz CPU clock is multiplied by a phase locked loop internal to the processor to achieve the 75 Mhz CPU speed. The processor contains 16 KB of internal cache and 256 KB of external cache on the logic board.

The 50 Mhz bus of the CPU is connected to a VL to PCI bridge chip from ACC microelectronics to generate the PCI bus. The bridge chip takes a 33.333 Mhz oscillator to make the PCI bus clock. The Cirrus Logic GD7542 video controller is driven from this bus and this bus has an external connector for future docking options.

The GD542 video controller has a 14.318 Mhz oscillator input which it uses internally to synthesize the higher video frequencies necessary to drive an internal 10.4" TFT panel or external CRT monitors. When running in VGA resolution modes the TFIT panel may be operated at the same time as the external analog monitor. For Super VGA resolutions only the external CRT may be used.

Operation input to portable computer **800** is made through the keyboard. An internal pointing device is imbedded in the keyboard. External connections are provided for a parallel device, a serial device, a PS/2 mouse or keyboard, a VGA monitor, and the expansion bus. Internal connections are made for a Hard Disk Drive, a Floppy Disk Drive, and additional memory.

Portable computer **800** contains 8 Megabytes of standard memory which may be increased by the user up to 32 Megabytes by installing optional expansion memory boards. The first memory expansion board can be obtained with either 8 or 16 Megabytes of memory. With the first expansion board installed another 8 Megabytes of memory may be attaches to this board to make the maximum amount.

A second serial port is connected to a Serial Infrared (SIR) device. This SIR device has an interface chip which uses a 3.6864 Mhz oscillator. The SIR port can be used to transmit serial data to other computers so equipped.

The two batteries of portable computer **800** are Lithium Ion and have internal controllers which monitor the capacity of the battery. These controllers use a 4.19 Mhz crystal internal to the battery.

Portable computer **800** has two slots for PCMCIA cards. These slots may be used with third party boards to provide various expansion options. Portable computer **800** also has an internal sound chip set which can be used to generate or record music and/or sound effects. An internal speaker and microphone built into the notebook. In addition, three audio jacks are provide for external microphones, audio input, and audio output.

FIG. **7** shows an exploded view of the TM5000TM made by Texas Instruments Incorporated. Table 2 describes the essential elements of FIG. **7**.

TABLE 2

Item	Description	Function
150	BASE	Base of computer
151	COVER ASSY,TOP	top cover of computer
154	CONNECTOR DOOR	connector door
155	PCMCIA DOOR	PCMCIA door
157	LCD ASSY,9.5"	compute display assembly
158	BEZBL,LCD	LCD display
160	Light Pipe	indicators for different functions (e.g. turbo mode)
161	BUTTON,BATTERY EJECT,LEFT	ejects left battery
162	BUTTON,BATTERY EJECT,RIGHT	ejects right battery
163	BUTTON,POWER SWITCH	power switch
166	HINGE COVER,RIGHT	hinge cover for display attachment to computer
167	BUTTON,PCM EJECT	PCMCIA eject buttons
168	HINGE COVER,LEFT	hinge cover for display attachment to computer
172	RAM CARD,FRONT TRIM	cover over ram card (ram cards not shown)
178	HINGE,RIGHT	hinge for attaching display to computer
179	HINGE,LEFT	hinge for attaching display to computer
181	HINGE,BRACKET,RIGHT	binge bracket for attaching display
182	HINGE,BRACKET,LEFT	hinge bracket for attaching display

TABLE 2-continued

Item	Description	Function
186	BRACKET,LEFT,FLOPPY DRIVE	bracket for floppy drive
187	LIGHT PIPE,HINGE COVER	indicators for different functions (e.g. power)
190	BRACKET,FLOPPY DRIVE	bracket for floppy drive
195	SPRING,I/O DOOR LATCH	latch for I/O doors
196	EXTENSION SPRING,I/O DOOR	extension spring for I/O doors
204	HEATSINK,CPU	heatsink for CPU
205	HEATSINK CUSHION	heatsink cushion
206	PWB ASSY,LED BOARD	printed wiring board for LEDs
210	PWB ASSY,MAIN BOARD	main printed circuit/wiring board
211	PWB ASSY,PCMCIA/SOUND BOARD	PCMCIA/Sound printed circuit/wiring board
212	PWB ASSY,KEYSCAN BD	keyscan printed circuit/wiring board
213	MICROFLOPPY DRIVE,11 MM	floppy drive
222	NAMEPLATE ,ACTIVE MATRIX COLOR	Nameplate
226	COVER,LCD SCREWS	screws for LCD
228	SCREW,TORX,PLASTITE,PAN,2–28 X .500	screws
229	SCREW,TORX,PLASTITE,4–20 X .250	screws
230	SCREW,TORX,SLOTTED,2–28 X.375",CARBON	screws
231	SCREW,TORX,MACHINE,BUTTON,2–56 X .1250	screws
232	SCREW,W/THREAD LOCK	screws
233	SCREW,SLOT-TORX,MACHINE,PAN,4–40 X .188	screws
234	SCREW,TORX,MACHINE,FLAT,4–40 X .375	screws
235	SCREW,METRIC,TORX,MACH,FLH,M3–0.5 X 6	screws
236	SCREW,TORX,4–20 X.375",CARBON STEEL	screws
237	SCREW,MACH,FLAT,PH,4–40 X .188	screws
238	SCREW,TORX,MACHINE,PAN,4–40 X .125	screws
239	SCREW,TORX,MACHINE,4–40 X .250	screws
240	SCREW,SLOT-TORX,PLASTITE,PAN,4–20 X 1.25	screws
241	SCREW,SLOT-TORX,PLASTITE,PAN,2–28 X .188	screws
242	SCREW,TORX,MACHINE,2–56 X .250	screws
243	SCREW,TORX,MACHINE,BUTTON,2–56 X .1875	screws
244	CABLE ASSY,LCD,RIGHT,W/O TAPE	cable
248	FLEX CABLE,HARD DISK DRIVE	flex cable
249	CABLE ASSY,FDD DX4	cable
253	CABLE EXTENSION MICROPHONE	cable for microphone
254	MEDALLION LABEL "P"	Texas Instruments trademark label
255	SECURITY RING	security ring
262	PWB ASSY,UNIVERSAL IR MODULE P/D	printed wiring board for IR module
263	LENS COVER,IR	lens cover for IR module
270	COMPRESSION FOAM,STANDBY SWITCH	foam for standby switch
271	BUTTON,STANDBY SWITCH SERIES	standby switch
275	Power input	input to computer from external power
276	Keyboard	Keyboard input

FIG. **8** shows an enlarged view of the main printed circuit board **210** of FIG. **7**. Note the CPU **204** and power input **275** are both on this printed circuit board **210**. The present invention can be implemented on the TM5000 by using the software control program, described herein, and the optional deadman timer circuit shown in FIG. **3**. The software control program would be run by the CPU **204** in memory (not shown) and communicate to the power switch. The optional deadman timer circuit would also be connected to the power switch **275** and the CPU **204** so that the deadman timer can be reset when necessary. The deadman timer circuit could be placed on the main printed circuit board **210**.

FIGS. **9–30** show logic diagrams of an implementation of the main printed circuit board **210** of the TM5000. This logic diagram details how the deadman timer circuit, and the logic for the shutdown procedure could be implemented, along with the other functions of a main printed circuit board.

FIGS. **31–35** show logic diagrams of an implementation of the keyscan printed circuit board **272** of the TM5000. This logic diagram details how the circuit could be designed to implement keyscan functions of the TM5000.

FIGS. **36–47** show logic diagrams of an implementation of the PCMCIA/Sound printed circuit board **211** of the TM5000. This logic diagram details how the circuit could be designed to implement keyscan functions of the TM5000.

FIGS. **48–49** show logic diagrams of an implementation of the IR module printed circuit board **262** of the TM5000. This logic diagram details how the circuit could be designed to implement infra-red module functions of the TM5000.

While several implementations of the preferred embodiment of the invention has been shown and described, various modifications and alternate embodiments will occur to those skilled in the art. For example, process diagrams are also representative of flow diagrams for microcoded and software based embodiments. In addition, various modifications and combinations of the illustrative embodiments, as well as other embodiments of the invention, will be apparent to persons skilled in the art upon reference to the description. Words of inclusion are to be interpreted as nonexhaustive in considering the scope of the invention. Various modifications and combinations of the illustrative embodiments, as well as other embodiments of the invention, will be apparent

to persons skilled in the art upon reference to the description. It is therefore intended that the appended claims encompass any such modifications or embodiments.

An example implementation of the software control program is included below. However, the invention could be implemented in a multitude ways and is not restricted to this implementation. In addition, the software control program includes calls to "FactoryPowerDownTable" and to "SubWalkTable" (hereafter referred to as WalkTables). These calls implement the shut down procedure of the invention. An example embodiment is included after the software control program. In this embodiment, devices are shut down in a specific order. However, the WalkTables may be altered to include a shut down procedure for other devices. For example, the WalkTables could shut down a real time clock, serial devices, floppy disk drives, hard disk drives, DMA controllers, interrupt controllers, and other peripheral devices on the main system bus. Further, the shut down procedure may include peripheral devices connected serially, or through the parallel port. In addition, the WalkTables could shut down peripheral devices on main system buses such as ESDI, AT, or PCI and may include devices on auxiliary buses, such as USB or 1394. Moreover, WalkTables could even shut down the entire bus itself. Furthermore, the WalkTables could even shut down portions of or the entire docking station that a portable computer may be connected to. These are just a few examples of what the shut down procedure could include and not meant to be an exhaustive listing. Various modifications and combinations of the illustrative shut down procedure, as well as other embodiments of the invention, will be apparent to persons skilled in the art upon reference to the description. It is therefore intended that the appended claims encompass any such modifications or embodiments.

```
;***************************************************************************
; ORIGINAL CODER: La Vaughn F. Watts, Jr.
;                                                   +------------------+
;+------------------------------------------------------ ¦ Near Entry Point
;¦ PmInit -- Initialize power management            +------------------+
;¦
;¦ Entry: None
;¦
;¦ Exit:  All registers preserved
;¦
;¦ This function is called during POST before executing INT 19h.
;+-------------------------------------------------------------------------+

public  PmInit
PmInit  proc   near
     assume  ds:nothing,es:nothing,fs:nothing,gs:nothing,ss:nothing

IFDEF   zzzlily                          ;Enable Smart Power Switch
        push     cx
        mov      cx,0
        in       al,0e2h
        and      al,NOT 7
        out      0e2h,al                 ;Kill software control
        loop     $
        or       al,02h
        out      0e2h,al
        loop     $

        or       al,1
        out      0e2h,al                 ;Smart power switch enabled
        pop      cx
ENDIF                                    ;zzzlily

;This code is called on exit of the PM initialization
;Restore all registers and quit.
;---------------------------------------------------------------------------
IFDEF  zzzlily                           ;Bring back the power switch
Quit:
        pop      gs
        pop      fs
        pop      es
        pop      ds
        popad
```

```
        popfd

AbortPmInit:
        Extrn   ExtCmosCsum:near
        Call    ExtCmosCsum


        mov     al,3bh
        out     70h,al
        in      al,71h
        in      al,0e1h
        and     al,01000000b
        cmp     al,0
        jne     quitswitch
        extrn   VidSuspend:near
        extrn   HDDDisable:near

        call    HDDDisable              ;Take hard drive Down!

        CLI

        in      al,0e2h
        and     al,NOT 3                ;5.08.1
        out     0e2h,al                 ;Smart disabled, power off if needed
        jmp     $

quitswitch:
        push    ax
        push    cx

Include Deadman.Inc                     ;Delays needed to
                                        ;initialize on powerup the R/C

        mov     cx,DEADMANDELAY

DeadmanPm1Delay:
        jmp     $+2
        loop    DeadmanPm1Delay


        in      al,0e0h
        or      al,01000000b            ;Turn on Power Switch SMI
        out     0e1h,al                 ;Clear interrupts
```

```
        mov     cx,DEADMANDELAY

DeadmanPm2Delay:
        jmp     $+2
        loop    DeadmanPm2Delay

        out     0e0h,al
        loop    $
;
;Is this a factory link? If so, leave Deadman timer off
;Otherwise turn it on!
;
        mov     ax,cs:BPVersion
        and     ah,0f0h             ;Test version?
        cmp     ah,0f0h             ;Maybe
        je      FactoryExit         ;NOP!

        in      al,0e2h
        or      al,04h              ;Turn it On-Deadman 5.07.01
        out     0e2h,al             ;Done

        mov     cx,DEADMANDELAY

DeadmanPm3Delay:
        jmp     $+2
        loop    DeadmanPm3Delay

FactoryExit:
        pop     cx
        pop     ax
        ret

ELSE                                ;zzzlily

Quit:   pop     gs
        pop     fs
        pop     es
        pop     ds
        popad
        popfd

AbortPmInit:
```

```
        ret
ENDIF                                   ;zzzlily

;File Deadman.inc
DEADMANDELAY    Equ  16+1               ;1.6 plus one for good measure
;Comments: 1000 works great..trying to min the delay now.
;Comments: so does 100,50,10
;Comments: Does NOT work: 5,8
;           ... remember we need delay that will work on 90MHz
;           and 120MHz, so put some margin in it.
;           120/75 =1.6 factor
;
;Code called from OS to turn off APM..this will terminate the smart switch
;this time for 15 seconds in order to orderly shutdown to new OS operating
;environment.

APMIFunc04      proc    near
IFDEF  zzzlily                          ;Arm Interrupts
;
;Reset Deadman Timer                    Gives MS 15 seconds to get back
;
        push    ax

        in      al,0e2h
        mov     ah,al                   ;5.08.1
        and     al,NOT 04

        out     0e2h,al                 ;Force to ZERO!
        mov     al,ah                   ;5.08.1
        out     0e2h,al                 ;5.08.1 Reset Complete

        pop     ax

        STI
ENDIF                                   ;zzzlily
                                        ;connection is established
        ret
APMIFunc04      endp

APMIFunc0B      proc    near
;This code is called by the OS every 1 to 5 seconds..welook at the event
;for the power switch and other power management devices.
```

```
;
IFDEF   zzzlily                                 ;Downcount Temperature
;------------------------------------------------------------------------
;Reset Deadman Timer                            Gives MS 15 seconds to get back
;
        push    cx
        in      al,0c2h
        or      al,4
        out     0e2h,al                         ;Force to one!

        mov     cx,DEADMANDELAY
Deadman1Delay:
        jmp     $+2
        loop    Deadman1Delay

        and     al,NOT 4
        out     0e2h,al                         ;Toggles to zero and leaves on NOW

        mov     cx,DEADMANDELAY

Deadman2Delay:
        jmp     $+2
        loop    Deadman2Delay

        or      al,4                            ;Resets to 15 seconds
        out     0e2h,al                         ;Done
        pop     cx

;Test for Wav/sound/IR active
;
;Scan for Unwanted SMI events
;
        pushf
        cli                                     ;Interrupts disabled
        in      al,0e1h                         ;Get interrupt mask please
        and     al, 0dfh                        ;make sure dock bit not set
                                                        ;dock bit means dock present, not an SMI
        cmp     al,0d3h                         ;Value of no interrupts pending
        je      KillUnwantedSmi                 ;Kill them out !
        xor     al,0d3h                         ;Flip the bits
        xchg    al,ah
        in      al,0e0h                         ;Mask active
        and     ah,al                           ;Only bits wanted
```

```
        jz      KillUnwantedSmi         ;Nothing logged

;
;We have an valid Smi posted, now we need to process!
;

IFDEF  zzzlilyp                         ;5.08.1 Force Software SMI

        mov     ah,59h
        in      al,0f2h                 ;Get old value
        xchg    al,ah
        out     0f2h,al                 ;Selected
        in      al,0f3h                 ;Old value
        or      al,10h                  ;Arm SMI
        out     0f3h,al                 ;Done

        mov     al,7fh
        out     0f2h,al
        in      al,0f3h                 ;Set up for low to high trans
        and     al, NOT 1
        out     0f3h,al                 ;Done

        or      al,1                    ;low to high, please

        STI                             ;Arm CPU interrupts
        out     0f3h,al                 ;Initiate the interrupt

        push    cx
        mov     cx,0                    ;Wait for interrupt
        loop    $
        loop    $
        loop    $
        loop    $
        loop    $
        pop     cx                      ;Interrupt complete

        CLI                             ;Setup for clean up
        mov     al,59h
        out     0f2h,al                 ;just in case, reset it
        in      al,0f3h
        and     al,NOT 10h              ;Kill the SMI
        out     0f3h,al                 ;Done
        xchg    al,ah
```

```
        out     0f2h,al                 ;Reset 0F2h
        CLI                             ;Disable again as we fixup
;
;We have processed the valid Smi posted!
;
KillUnwantedSmi:
        mov     ax,01059h               ;Kill the SMI
        Extrn   CfgClearBits:near
        call    CfgClearBits            ;Done
        mov     ax,017dh
        call    CfgClearBits            ;Clr the interrupt
        popf                            ;Clean up stack
ENDIF                                   ;zzzlilyp

IFDEF  zzzlilyd                         ;Is it a Lilyd?-Yes, then add SoftSMI
;                                       ;Yes, we must add SMI software
;We have processed the valid Smi posted!
;

KillUnwantedSmi:
        popf                            ;Clean up stack

ENDIF                                   ;zzzlilyd

;
;We are now free of that issue, at lease for now!
;
        mov     al,38h                  ;Value with Sound/IR active
        call    CmosRead                ;Read it
        test    ah,08h                  ;Sound/IR bit ON = Active
        jnz     NoFunc08PCChange        ;Do not compute this time

        extrn   DoThermalManagement:near
        extrn   BPPowerChange:near

        mov     al,7ah                  ;Get the seconds passed for test
        call    CmosRead                ;Done
        test    ah,40h                  ;Get Battery Capacity Request active
        jz      SkipBatteryCap          ;Skip this request
;
;We need a read for the battery capacity
;
```

```
        Extrn   ReadLilyBattery:near
        Call    ReadLilyBattery
        jnc     HaveBatteryData         ;Data is good, update it
;
;BX = 0, No channel available, use previous data
;BX = 2, No batteries/battery installed, tell user
;
        cmp     bx,0
        je      NoFunc08PCChange        ;Bad data point
;
;Setup unknown for user
;
        mov     al,7fh                  ;Unknown?

HaveBatteryData:
;
;Data good;AL = Percent available, ch = status slot a cl =status slot b
;
        mov     ah,39h
        xchg    al,ah
        mov     bl,7fh                  ;Mask to write
        call    CmosWriteMask           ;Done and its good!

;Protect against the power switch being turned off during update
;
        xchg    bx,ax                   ;
        pushf                           ;
        cli                             ;Disable interrupts
        in      al,0e2h                 ;Get software status
        mov     ah,al                   ;
        or      al,3                    ;Force to software override
        out     0e2h,al                 ;
        xchg    ax,bx                   ;Read to write data
        mov     ax,007ah
        mov     bl,40h
        call    CmosWriteMask           ;Reset the Request Bit
        Call    ExtCmosCsum

        xchg    ax,bx                   ;
        xchg    ah,al                   ;
        out     0e2h,al                 ;Put switch back to way it was
        popf                            ;Restore interrupts to way it was
```

```
        jmp     short NoFunc08PCChange

SkipBatteryCap:
        mov     bl,ah                   ;Power status flag
        and     ah,3fh                  ;Number of minutes
        and     bl,80h                  ;PowerChange flag
;
;Look at power change status
;
        in      al,0e3h                 ;Port containing AC information
        and     al,00001000b
        shl     al,4                    ;Align with old value
        dec     ah
        cmp     ah,0                    ;Time to read?
        jne     ThermalTest1            ;Not yet
        mov     ah,63                   ;63= seconds for a minutes;best we
        call    DoThermalManagement;can do and allow up to 3 battery
                                        ;reads during cycle!
ThermalTest1:
        cmp     al,bl                   ;PowerChange state occured?
        je      ThermalTest2            ;Nop
        call    BPPowerChange           ;Execute the power state change code
        or      ah,40h                  ;Force battery read once a DoPowerC

ThermalTest2:
        and     ah,7fh
        or      ah,al                   ;New value to write
        mov     al,7ah

;
;Protect against the power switch being turned off during update
;
        xchg    bx,ax
        pushf
        cli                             ;Disable interrupts
        in      al,0e2h                 ;Get software status
        mov     ah,al                   ;
        or      al,3                    ;Force to software override
        out     0e2h,al                 ;
        xchg    ax,bx                   ;Read to write data
        Extrn   CmosWrite:near
        Call    CmosWrite
        Extrn   ExtCmosCsum:near
```

```
        Call    ExtCmosCsum

        xchg    ax,bx
        xchg    ah,al
        out     0e2h,al                 ;Put switch back to way it was
        popf                            ;Restore interrupts to way it was
;
;End of protecting power switch
;

NoFunc08PCChange:
;...return to OS here...
ENDIF                                   ;zzzlily

;this code can be called from DOS and power management and thermal
;management vectors anytime.
BPPowerCheck:                           ;Please call only in DOS
        in      al,0e1h                 ;Get PS State
        and     al,01000000b            ;PowerSwitch State
        jne     BPNoFuncSupport                 ;Switch is NOT to "turn it off"

;turn off if switch is off
;this code can be left in if docking station does not have power switch intelligence
;
;       in      al, 0e1h                ;switch off, check to see if we be docked
;       test    al, 020h                ;
;       jnz     BPNoFuncSupport         ;ignore off position if docked

;
;Insert the walk tables here
;
        in      al,0e2h                 ;Deadman active?


;following code is optional and provides software override for windows.
;If it is on now, power off...
;       and   al,04h                    ;Maybe
;       jne   BPNoFuncSupport           ;Yes, Skip the power down

        in      al,0e0h                 ;Mask wanted?
        and     al,01000000b
        je      BPNoFuncSupport                 ;Nop, skip this request
```

```
        extrn   FactoryPowerDownTable:byte
        mov     si, offset FactoryPowerDownTable

        extrn   SubWalkTable:near
        call    SubWalkTable
        jmp     short BPGoDownDownDown

BPPowerDown:                            ;Factory turnpower off
;
;Insert the walk tables here
;
        extrn   FactoryPowerDownTable:byte
        mov     si, offset FactoryPowerDownTable

        call    SubWalkTable

BPGoDownDownDown:

        Extrn   TurnPowerOff:near
        in      al,0e2h                 ;Get power switch to clr
        and     al,11111000b            ;Turn it off
        or      al,00000011b            ;Turn it on
        out     0e2h,al
        loop    $

        Call    TurnPowerOff
        jmp     BPNoFuncSupport

BPGetVersion:
        mov     bx,cs:word ptr BPVersion
        mov     ax,cs:word ptr BPRevision
        STI
        clc
        ret

;This code lands here if an interrupt has happened by the SMI.
;We look to see if power switch has been hit..if smart power switch is
;programmed to be "Save-to-Disk" we generate the Save to disk event.
;Smart power switch is set to "suspend or standby" then we can generate
;the suspend or standby event. All events are transferred to OS for
;final operation where they turn around and call APMIFunction 07 to
;turn off power after cleaning up or saving to disk. If shutdown (super
;one) is running, then we pass the event to him for file savings and task
```

```
;closings prior to giving the event to OS.
;Note: APMIFunction07 is the same as the turnpoweroff code above.
;Note: there are two ways that we support this smartpower switch for the
;user control;
;1. Let the hardware map the interrupt of the smart power switch to the
;suspend, standby, donothing, save-to-disk, turn power off event.
;2. Read one common event and let the software read cmos map that contains
;the user setup option and map it after we get the same hardware event.
;
SMI_Interrupt:
;
;Test for power switch turning off - timing issue
;
        test    ah,40h          ;Check for AC/PowerSwitch request
        jz      PPLatch         ;Process Powerswitch : bit = 0
        test    ah,8            ;Check for Suspend Key
        jnz     PSKey           ;Process Suspend Key : bit =1
        test    ah,4            ;Check Stby Key
        jnz     PStbyKey        ;Process Stby Key : bit =1
        test    ah,2            ;Check Closed Cover latch
        jz      PCLatch         ;Process closed cover latch : bit =1
        test    ah,1            ;Check low battery alarm
        jz      PBLatch         ;Process low battery alarm : bit =0
        test    ah,20h          ;Check Dock/Undock Request
        jnz     PDLatch         ;Process Dock/Undock Request : bit =0
        test    ah,40h          ;Check for AC/PowerSwitch request
        jz      PPLatch         ;Process Powerswitch : bit = 0
        test    ah,10h          ;Check for EZ-dock com request
        jz      PELatch         ;Process EZ-Dock com : bit =1
                                ;No more request, just in case,
        jmp     PDLatch         ;Process apparent Dock/Undock Request

        in      al,0e0h         ;Interrupt accept Mask
        out     0e1h,al         ;clr all since we should not be here.

NoActionKey:
        mov     ax, 0107dh      ;Clear status Using EXTSMI0
        call    CfgClearBits    ;
        RET                     ;Finished here!
;
;Process routines
;
PSKey:                          ;Suspend Key
```

```
                                        ;ah&al=E1h
        mov     al,8                    ;Clr interrupt
        out     0e1h,al                 ;Done
        mov     ax, 0107dh              ;Clear status Using EXTSMI0
        call  CfgClearBits              ;

SusLBatAction:
        mov     al,5ch                  ;Get suspend key action needed
        mov     bl,11000000b            ;Get options
SusAction:
        call    CmosReadMask
        cmp     ah,0
        je      NoActionKey             ;Ignore this key
        cmp     ah,2                    ;Stby wanted?
        je      ActionStby              ;Yep
        cmp     ah,1
        je      ActionSus               ;Suspend action
        jmp     SaveDiskAction          ;Save-to-Disk
PStbyKey:

        mov     al,4                    ;Clr interrupt
        out     0e1h,al                 ;Done
        mov     ax, 0107dh              ;Clear status Using EXTSMI0
        call    CfgClearBits            ;

ActionStby:
        TREPORT088h
;
;Test for Wav/sound/IR active
;
        mov     al,38h                  ;Value with Sound/IR active
        call    CmosRead                ;Read it
        test    ah,08h                  ;Sound/IR bit ON = Active
        jnz     NoActionKey             ;Do not compute this time

        JMP     GlobalStby              ;Do it

ActionSus:
        TREPORT                         084h
;
;Test for Wav/sound/IR active
;
        mov     al,38h                  ;Value with Sound/IR active
```

```
        call    CmosRead            ;Read it
        test    ah,08h              ;Sound/IR bit ON = Active
        jnz     NoActionKey         ;Do not compute this time

        JMP     GlobalSus           ;Do it

Public  PCLatch
PCLatch:                            ;Process closed cover latch : bit =1
        extrn   Video_Global:near
        extrn   Video_UnGlobal:ncar
;
;Note: Lowtime 40:6c words
;       40:6e words

        push    cx
        mov     cx,7                ;Number of seconds to delay/3
        extrn   KeyDisable:ncar
        extrn   KeyEnable:near
        extrn   WaitSecDelay:near

StallPCLatch:
        call    KeyDisable
        call    WaitSecDelay        ;Wait one second

        in      al,0e1h             ;Read the Cover latch & Low Bat
        test    al,2                ;Still down?
        jnz     PCLatchAbort        ;Nop, abort the saving status
        test    al,40h
        jz      PPLatch             ;turn power off
        call    KeyEnable
        call    WaitSecDelay        ;Wait one second
        in      al,0e1h             ;Read the Cover latch & Low Bat
        test    al,2                ;Still down?
        jnz     PCLatchAbort        ;Nop, abort the saving status
        test    al,40h
        jz      PPLatch             ;turn power off
        loop    StallPCLatch

CCLBDoit:
        in      al,0e1h             ;Read the Cover latch & Low Bat
        test    al,2                ;Still down?
        jnz     PCLatchAbort        ;Nop, abort the saving status
        test    al,40h
```

```
        jz      PPLatch             ;turn power off
        call    Keydisable
        pop     cx                  ;Clean Stack off
        mov     al,2
        out     0e1h,al             ;Clear Interrupt - Both!
        mov     al,5ch              ;Get suspend key action needed
        mov     bl,00110000b        ;Get options
        jmp     SusAction           ;Process based on user

Public PCLatchAbort
PCLatchAbort:
        Call    KeyEnable
        mov     ah,5ah              ;Read current status
        call    CfgRead             ;Get the value
        and     ah,20h              ;Alarms on?
        cmp     ah,20h              ;maybe
        jne     PCLatchAb1          ;Nop
        mov     ax,205ah
        call    CfgClearBits        ;clear alarm suspend request
        xor     cx,cx
        loop    $
        mov     ax,205ah
        call    CfgSetBits
        xor     cx,cx
        loop    $

PCLatchAb1:
        pop     cx                  ;Clean Stack off
        mov     al,2

        out     0e1h,al             ;Clear Interrupt
        mov     ax, 0047dh          ;Clear status
        call    CfgClearBits

        jmp     NoActionKey

PBLatch_Clr:                        ;low battery while docked
        mov     al,001h
        out     0e1h,al             ;Clear Interrupt
        jmp     NoActionKey

;
;We have standby here
```

```
;

PBLatch:                                    ;Process low battery alarm : bit =0
        call    APMBattLowNotify    ;Tell APM

        push    cx
        mov     cx,4                ;Number of seconds to delay/2

StallPBLatch:
        call    KeyDisable
        call    WaitSecDelay        ;Wait one second

        in      al,0e1h             ;Read the Cover latch & Low Bat
        test    al,1                ;Still down?
        jnz     PBLatchAbort        ;Nop, abort the saving status

        test    al,40h
        jz      PPLatch             ;turn power off

        call    KeyEnable
        call    WaitSecDelay        ;Wait one second

        in      al,0e1h             ;Read the Cover latch & Low Bat
        test    al,1                ;Still down?
        jnz     PBLatchAbort        ;Nop, abort the saving status

        test    al,40h
        jz      PPLatch             ;turn power off

        loop    StallPBLatch

LBDoit:
        in      al,0e1h             ;Read the Cover latch & Low Bat
        test    al,1                ;Still down?
        jnz     PBLatchAbort        ;Nop, abort the saving status

        test    al,40h
        jz      PPLatch             ;turn power off

        call    Keydisable

        pop     cx                  ;Clean Stack off
        mov     al,1
```

```
        out     0e1h,al              ;Clear Interrupt - Both!
        mov     al,5ch               ;Get suspend key action needed
        mov     bl,00110000b         ;Get options
        jmp     SusAction            ;Process based on user

Public PBLatchAbort
PBLatchAbort:
        Call    KeyEnable

        mov     ah,5ah               ;Read current status
        call    CfgRead              ;Get the value
        and     ah,20h               ;Alarms on?
        cmp     ah,20h               ;maybe
        jne     PBLatchAb1           ;Nop
        mov     ax,205ah
        call    CfgClearBits         ;clear alarm suspend request
        xor     cx,cx
        loop    $
        mov     ax,205ah

        call    CfgSetBits
        xor     cx,cx
        loop    $

PBLatchAb1:
        pop     cx                   ;Clean Stack off

        mov     al,1
        out     0e1h,al              ;Clear Interrupt

        mov     ax, 0047dh           ;Clear status
        call    CfgClearBits

        jmp     NoActionKey

PDLatch:                                     ;Process Dock/Undock Request : bit =0
        in      al, 0e1h             ;are we docked?
        test    al, 020h
        jz      NotDocked

Docked:
        in      al, 0e0h             ;disable closed cover SMI
        and     al, 0fdh
```

```
        out     0e0h, al
        jmp     @f

NotDocked:
        in      al, 0e3h            ;are we AC power?
        test    al, 08h
        jnz     @f                  ;yes, don't worry about cover
        in      al, 0e0h            ;else, enable closed cover SMI
        or      al, 02h
        out     0e0h, al
@@:
        mov     al, 020h            ;clear dock/undock SMI
        out     0e1h, al

        jmp     NoActionKey

PPLatch_Chng:                           ;power switch changed while docked
        mov     ah, 098h            ;switch is on flag - debug stuff
        in      al, 0e1h            ;check power switch position
        and     al, 040h
        jnz     @f
        in      al, 0e3h            ;check for ac power
        and     al, 08h
        jz      PPLatch2            ;turn off if on battery
        jmp     PStbyKey            ;else go into Standby
        mov     ah, 099h            ;switch is off flag - debug stuff
@@:
        mov     al, 040h            ;clear power switch SMI
        out     0e1h, al
        jmp     NoActionKey

PPLatch:                            ;Process Powerswitch : bit = 0
@@:                                     ;we will process power switch

PPLatch2:
        mov     al,0
        out     0e0h,al             ;Kill all interrupts
        mov     al,-1
        out     0e1h,al             ;Clr all pending ones

;
;Insert the walk tables here
;
```

```
        extrn   PowerDownTable:byte
        mov     si, offset PowerDownTable
        extrn   SubWalkTable:near
        call    SubWalkTable

Public  TurnPowerOff
TurnPowerOff    proc    near
        Extrn   ExtCmosCsum:near
        Call    ExtCmosCsum

turnpwroff:
        CLI                             ;Disable interrupts
        mov     al,0
        out     0e0h,al                 ;Kill all interrupts
        mov     al,-1
        out     0e1h,al                 ;Clr all pending ones

        in      al,0e2h
        and     al,NOT 4                ;Turn off Power Deadman
        out     0e2h,al                 ;Done

        or      al,7                    ;Turn on Software Control/Deadman
        out     0e2h,al                 ;Done

        and     al,NOT 3                ;Turn off software Control
        out     0e2h,al

        or      al,1                    ;Turn power off please
        out     0e2h,al                 ;Done!
Forever: JMP    Forever                 ;Spin until loss of power or deadman control
TurnPowerOff    endp


PELatch:                                ;Process EZ-Dock com : bit =1
        in      al,0e0h
        out     0e1h,al                 ;clear interrupt
        mov     ax, 0017dh              ;Clear status Using EXTSMI0
        call    CfgClearBits

        jmp     NoActionKey

SaveDiskAction:
        extrn   SaveToDisk:near
```

```
        call        SaveToDisk
;
;Need to add critical resume to que if Windows 95
;
        mov         al,0ffh
        out         0e1h,al
endif                                   ;zzzlilly
        ret
APMIFunc0b  endp




;*******************************************************************************
;Beginning of Walktable examples:
;*******************************************************************************

      extrn  Com_Lpt_Suspend:near        ;COM, LPT suspend code
      extrn  Com_Lpt_Resume:near         ;COM, LPT resume code

      extrn  Floppy_Suspend:near         ;FLOPPY suspend code
      extrn  Floppy_Resume:near          ;FLOPPY resume code
      extrn  FPU_Suspend:near            ;Coprocessor suspend code
      extrn  FPU_Resume:near             ;Coprocessor resume code
      extrn  Keyboard_Suspend:near       ;KBD suspend code
      extrn  Keyboard_Resume:near        ;KBD resume code

      extrn  Video_Global:near           ;VIDEO panel off and suspend code
      extrn  Video_UnGlobal:near         ;VIDEO panel on and resume code

      extrn  MarkUTime:near
IFDEF  zzzlily                           ;Extermal for Features SuspendVideo
      extrn  VidSuspend:near             ;Video___P
      extrn  SetAutoSuspendTimer:near
      extrn  ClrAutoSuspendTimer:near
      extrn  HDDDisable:near
      extrn  HDDUp:near
      extrn  KeyEnable:near
      extrn  KeyDisable:near
      extrn  PCISleep:near
      extrn  PCIInit:near
```

```
        extrn  SuspendInitialize:near
        extrn  ClrActivityTimer:near
        extrn  DozeInitialize:near
ENDIF  ;zzzlily

;
; SUSPENDTable - User may alter the sequence in the SUSPENDTable, also,
;            user may add customization code in this SUSPENDTable.
;            For example, VPx_Suspend is to set value to power
;            register 0/1 to turn on/off devices that connected to
;            power register 0/1.
;
     public  SuspendTable
SuspendTable   label  word

IFDEF  zzzlily                           ;Suspend Table - Devices
        dw     offset  ClrAutoSuspendTimer
        dw     offset  ClrActivityTimer
        dw     offset VidSuspend          ;Place Screen Memory into suspend
        dw     offset HDDDisable          ;Take hard drive Down!
        dw     offset FPU_Suspend         ;Coprocessor suspend now, 6th executed
        dw     offset SirOff              ;Sleep SIR Leds
        dw     WaitSecDelay               ;Debug
        dw     offset KeyDisable
        dw     WaitSecDelay               ;Debug
        dw     offset PCISleep            ;Sleep PCI bus
ENDIF  ;zzzlily
        dw     END_TABLE                  ;End of table, do not add anything
                                          ;here
;
; RESUME Functions - Resume table, please see SUSPENDTable for references.
;
     public  ResumeTable
ResumeTable   label  word
IFDEF  zzzlily                            ;Add PCIInit
        extrn  WaitSecDelay:near          ;Debug

;  This table was modified by Ashish Hira.  The table was
;  rearraged to solve a problem when resuming from suspend.  Trouble
;  report reference number 2897.
;
;  Description:
;  External PS2 mouse erratic sometimes, but the internal works when
```

```
; coming out of suspend or standby.
;
; Found out - the sequence of resuming had a impact on keyboard being
; enable for the first few seconds when resuming. If the user moved the
; external mouse in that time, the mouse goes erratic.

        dw      offset PCIInit              ; Bring up PCI Bus
        dw      offset MarkUTime            ;
        dw      offset FPU_Resume           ;
        dw      offset SuspendInitialize;
        dw      offset KeyEnable            ; Use Keyboard from BatteryPro
        dw      offset Video_Unglobal       ;
        dw      offset SirOn                ;

ENDIF ;zzzlily
        dw      END_TABLE
;
; GLOBAL Functions - Global Standby table, please see SUSPENDTable for
;               references.
;
    public GlobalTable
GlobalTable   label  word
IFDEF  zzzlily                              ;Global Suspend table
        dw      offset Video_Global
        dw      offset SetAutoSuspendTimer
        dw      END_TABLE
ENDIF  ;zzzlily
IFDEF  zzzlily                              ;PowerChange Table
    Public  PChangeTable
PChangeTable   label  word
        extrn   SoundTVInitialize:near
        dw      SoundTVInitialize           ;vw-done;Setup Sound/TV modes
        dw      PCIInit                     ;Initialize PCI bus
        extrn   HDSetTim:near
        dw      HDSetTim                    ;vw-done;Initialize HDD Timeouts
        extrn   LocalInitialize:near
        dw      LocalInitialize
        extrn   GlobalInitialize:near
        dw      GlobalInitialize
        dw      SuspendInitialize
        dw      END_TABLE
ENDIF  ;zzzlily
;
```

```
; UNGLOBAL Functions - Exit Global Standby table, please see SUSPENDTable
;               for references.
;
     public  ExitGlobalTable
ExitGlobalTable label   word
IFDEF  zzzlily                          ;Add PCIInit
        dw     PCIInit                  ;Bring up PCI Bus
        dw     SuspendInitialize
        dw     offset Video_UnGlobal
        dw     END_TABLE

ENDIF  ;zzzlily
        dw     END_TABLE
IFDEF  zzzlily                          ;Smart Pwer switch table
;
; POWERDOWNTABLE - User may alter the sequence in the POWERDOWNTable, also,
;           user may add customization code in this Table.
;           For example, VPx_Suspend is to set value to power
;           register 0/1 to turn on/off devices that connected to
;           power register 0/1.
;
public  PowerDownTable
PowerDownTable   label   word
        dw     offset VidSuspend        ;Place Screen Memory into suspend
        dw     offset HDDDisable        ;Take hard drive Down!
        dw     END_TABLE                ;End of table, do not add anything

public  FactoryPowerDownTable
FactoryPowerDownTable   label   word
        dw     offset HDDDisable        ;Take hard drive Down!
        dw     END_TABLE                ;End of table, do not add anything

ENDIF  ;zzzlily
```

What is claimed is:

1. An apparatus, comprising:

a means for user input;

a means for output;

a processor coupled to said means for user input and means for output;

a software controlled switch for coupling power to said processor, said switch having a first mode of operation wherein power to said processor is terminated substantially simultaneously with user actuation of said switch, and a second mode of operation wherein power to said processor is terminated upon completion of both said switch being user actuated and software releasing control of said switch at a time not substantially simultaneous with user actuation of said switch; and

a timer circuit coupled to said switch, said timer circuit including a power off timer with a set value that initiates a shut down procedure when said power off timer times out.

2. The apparatus of claim 1, wherein said software controlled switch further includes a third mode of operation wherein power to said processor is terminated upon completion of said switch being user actuated and one of said software releasing control of said switch within a time period less than said set value and said power off timer completing said shut down procedure in the event said software does not release control of said switch within a time period less than said set value.

3. The apparatus of claim 2, wherein said software sets said power off timer to time out instantly upon completion of both said switch being user actuated and said software releasing control of said switch prior to said time period reaching said set value.

4. The apparatus of claim 1, wherein said software controlling said switch in said second mode of operation initiates a software shutdown process.

5. The apparatus of claim 1, wherein said software controlling said switch in said second mode of operation initiates a hardware shutdown process.

6. The apparatus of claim 1, wherein said software controlling said switch in said second mode of operation initiates a software and hardware shutdown process.

7. The apparatus of claim 1, wherein said set value is a default value.

8. The apparatus of claim 1, wherein said software sets said default value.

9. The apparatus of claim 1, wherein a user of said apparatus sets said default value.

10. The apparatus of claim 1, wherein said set value is resetable by said processor.

11. The apparatus of claim 1, wherein said processor is a central processing unit (CPU).

12. The apparatus of claim 1, wherein said processor is an application processor.

13. The apparatus of claim 1, wherein said apparatus is a computer.

14. The computer of claim 13, wherein said software is implemented in the bootup process of said computer.

15. The computer of claim 13, wherein said software is implemented when a user of said computer actuates said switch.

16. The computer of claim 13, wherein said software controlled switch further includes a power failure mode wherein a power shut down procedure is initiated in the event that battery power level drops below a predetermined value.

17. The computer of claim 13, wherein when said power switch is actuated to be on, the system boots, the software boots, the Basic Input/Output System (BIOS) initializes and then the timer is set to zero and the power switch is activated in said first mode of operation.

18. The computer of claim 17, wherein subsequent to said power switch being activated in said first mode of operation, and through the process of initializing the rest of the computer's system, said software determines whether or not to change said switch from said first mode of operation to said second mode of operation.

19. The computer of claim 13, wherein said switch is programmed to watch for a System Management Interrupt (SMI).

20. The computer of claim 19, wherein said switch is set to act real time upon an SMI.

21. The computer of claim 20, wherein when the SMI interrupt is detected, the heads of a hard drive coupled to said processor are positioned and parked, the power to the hard drive and a display coupled to the processor is terminated, after which the CMOS parameters that need to be saved are saved.

22. The computer of claim 13, wherein said switch is set to act upon an SMI at a later time.

23. The computer of claim 22, wherein said software allows the computer's operating system and other programs to prepare for shut down, including but not limited to, closing files, updating any pertinent parameters, after which the heads of a hard drive coupled to said processor are positioned and parked, the power to the hard drive and a display coupled to the processor is terminated, after which the CMOS parameters that need to be saved are saved.

24. A computer, comprising:

a display;

a keyboard;

a central processor unit (CPU) coupled to said display and said keyboard;

a software controlled switch for coupling power to said central processing unit (CPU), said switch having a first mode of operation wherein power to said processor is terminated substantially simultaneously with user actuation of said switch, and a second mode of operation wherein power to said central processing unit (CPU) is terminated upon completion of both said switch being user actuated and software releasing control of said switch at a time not substantially simultaneous with user actuation of said switch; and

a timer circuit coupled to said switch, said timer circuit including a power off timer with a set value that initiates a shut down procedure when said power off timer times out.

25. The computer of claim 24, wherein said software controlled switch further includes a third mode of operation wherein power to said central processing unit (CPU) is terminated upon completion of said switch being user actuated and one of said software releasing control of said switch within a time period less than said set value and said power off timer completing said shut down procedure in the event said software does not release control of said switch within a time period less than said set value.

26. The computer of claim 25, wherein said software sets said power off timer to time out instantly upon completion of both said switch being actuated and said software releasing control of said switch prior to said time period reaching said set value.

27. The computer of claim 24, wherein said software controlling said switch in said second mode of operation initiates a software shutdown process.

28. The computer of claim 24, wherein said software controlling said switch in said second mode of operation initiates a hardware shutdown process.

29. The computer of claim 24, wherein said software controlling said switch in said second mode of operation initiates a software and hardware shutdown process.

30. The computer of claim 24, wherein said set value is a default value.

31. The computer of claim 24, wherein said software sets said default value.

32. The computer of claim 24, wherein a user of said apparatus sets said default value.

33. The computer of claim 24, wherein said set value is resetable by said central processing unit (CPU).

34. A method of controlling power to an apparatus, comprising the step of:

providing a user input;

providing an output;

providing a processor coupled to said user input and output;

providing a software controlled switch for coupling power to said processor, said switch having a first mode of operation wherein power to said processor is terminated substantially simultaneously with user actuation of said switch, and a second mode of operation wherein power to said processor is terminated upon completion of both said switch being user actuated and software releasing control of said switch at a time not substantially simultaneous with user actuation of said switch; and

providing a timer circuit for coupling to said switch, said timer circuit including a power off timer with a set value that initiates a shut down procedure when said power off timer times out.

35. The method of claim 34, wherein said software controlled switch further includes a third mode of operation wherein power to said processor is terminated upon completion of said switch being user actuated and one of said software releasing control of said switch within a time period less than said set value and said power off timer completing said shut down procedure in the event said software does not release control of said switch within a time period less than said set value.

36. The method of claim 35, wherein said software sets said power off timer to time out instantly upon completion of both said switch being user actuated and said software releasing control of said switch prior to said time period reaching said set value.

37. The method of claim 34, wherein said software controlling said switch in said second mode of operation initiates a software shutdown process.

38. The method of claim 34, wherein said software controlling said switch in said second mode of operation initiates a hardware shutdown process.

39. The method of claim 34, wherein said software controlling said switch in said second mode of operation initiates a software and hardware shutdown process.

40. The method of claim 34, wherein said set value is a default value.

41. The method of claim 34, wherein said software sets said default value.

42. The method of claim 34, wherein a user of said apparatus sets said default value.

43. The method of claim 34, wherein said set value is resetable by said processor.

44. The method of claim 34, wherein said processor is a central processing unit (CPU).

45. The method of claim 34, wherein said processor is an application processor.

46. The method of claim 34, wherein said apparatus is a computer.

47. The method of claim 46, wherein said software is implemented in the bootup process of said computer.

48. The method of claim 46, wherein said software is implemented when a user of said computer actuates said switch.

49. The method of claim 46, wherein said software controlled switch further includes a power failure mode wherein a power shut down procedure is initiated in the event that battery power level drops below a predetermined value.

50. The method of claim 46, wherein when said power switch is actuated to be on, the system boots, the software boots, the Basic Input/Output System (BIOS) initializes and then the timer is set to zero and the power switch is activated in said first mode of operation.

51. The method of claim 50, wherein subsequent to said power switch being activated in said first mode of operation, and through the process of initializing the rest of the computer's system, said software determines whether or not to change said switch from said first mode of operation to said second mode of operation.

52. The method of claim 51, wherein said software allows the computer's operating system and other programs to prepare for shut down, including but not limited to, closing files, updating any pertinent parameters, after which the heads of a hard drive coupled to said processor are positioned and parked, the power to the hard drive and a display coupled to the processor is terminated, after which the CMOS parameters that need to be saved are saved.

53. The method of claim 46, wherein said switch is programmed to watch for a System Management Interrupt (SMI).

54. The method of claim 53, wherein said switch is set to act real time upon an SMI.

55. The method of claim 54, wherein when the SMI interrupt is detected, the heads of a hard drive coupled to said processor are positioned and parked, the power to the hard drive and a display coupled to the processor is terminated, after which the CMOS parameters that need to be saved are saved.

56. The method of claim 46, wherein said switch is set to act upon an SMI at a later time.

57. An apparatus, comprising:

a means for user input;

a means for output;

a processor coupled to said means for user input and means for output;

circuitry for coupling power to said processor, said circuitry having a first mode of operation wherein power to said processor is terminated substantially simultaneously with user actuation of a switch, and a second mode of operation wherein power to said processor is terminated upon completion of both said switch being user actuated and software releasing control of said circuitry at a time not substantially simultaneous with user actuation of said switch; and

a timer circuit coupled to said circuitry, said timer circuit including a power off timer with a set value that initiates a shut down procedure when said power off timer times out.

58. An apparatus, comprising:

a means for user input;

a means for output;

a processor coupled to said means for user input and means for output;

a software program executed on said processor, said software program facilitating a first mode of operation wherein power to said processor is terminated substantially simultaneously with user actuation of a switch, and a second mode of operation wherein power to said processor is terminated upon completion of both said switch being user actuated and said software program

triggering said termination of power at a time not substantially simultaneous with user actuation of said switch; and

a timer function for initiating a termination of power to said processor when said timer times out.

59. The apparatus of claim 58, wherein said timer function is performed by hardware.

* * * * *

EXHIBIT 2

US006173409B1

(12) **United States Patent**
Watts, Jr. et al.

(10) **Patent No.:** **US 6,173,409 B1**
(45) **Date of Patent:** **Jan. 9, 2001**

(54) **REAL-TIME POWER CONSERVATION FOR ELECTRONIC DEVICE HAVING A PROCESSOR**

(75) Inventors: **LaVaughn F. Watts, Jr.**; **Steven J. Wallace**, both of Temple, TX (US)

(73) Assignee: **Texas Instruments Incorporated**, Dallas, TX (US)

(*) Notice: Under 35 U.S.C. 154(b), the term of this patent shall be extended for 0 days.

(21) Appl. No.: **09/392,205**

(22) Filed: **Sep. 8, 1999**

Related U.S. Application Data

(63) Continuation of application No. 08/023,831, filed on Apr. 12, 1993, now Pat. No. 6,006,336, which is a continuation of application No. 07/429,270, filed on Oct. 30, 1989, now Pat. No. 5,218,704.

(51) **Int. Cl.**[7] **G06F 1/32**

(52) **U.S. Cl.** **713/322**; 713/601

(58) **Field of Search** 713/322, 601, 713/320, 300, 600

(56) **References Cited**

U.S. PATENT DOCUMENTS

3,453,601	7/1969	Bogert et al. .
3,623,017	11/1971	Lowell .
3,868,647	2/1975	Zandvied .
3,922,526	11/1975	Cochran .
3,941,989	3/1976	McLaughlin et al. .
4,137,563	1/1979	Tsunoda .
4,217,637	8/1980	Faulkner et al. .
4,254,475	3/1981	Cooney et al. .
4,267,577	5/1981	Hashimoto et al. .
4,279,020	7/1981	Christian et al. .
4,293,927	10/1981	Hoshii .
4,316,247	2/1982	Iwamoto .
4,317,180	2/1982	Lies .
4,317,181	2/1982	Teza et al. .
4,361,873	11/1982	Harper et al. .
4,381,552	4/1983	Nocilini et al. .
4,409,665	10/1983	Tubbs et al. .
4,590,553	5/1986	Noda .
4,612,418	9/1986	Takeda et al. .
4,615,006	9/1986	Maejima et al. .
4,670,837	6/1987	Sheets .

(List continued on next page.)

FOREIGN PATENT DOCUMENTS

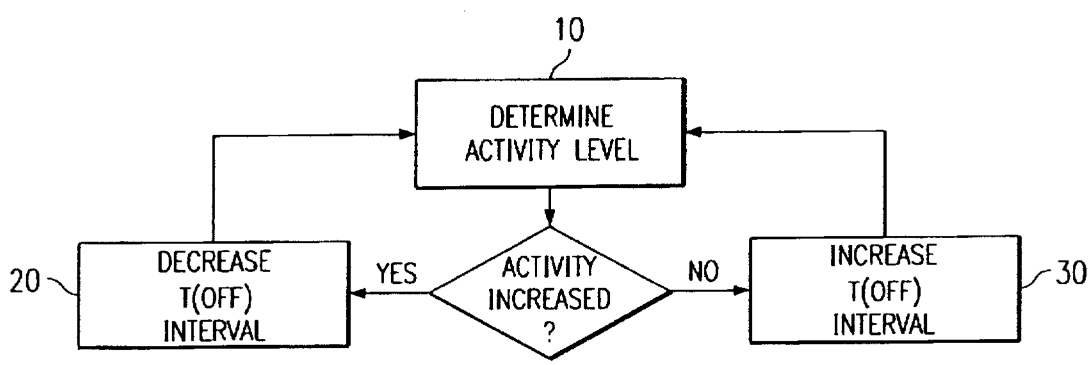
0 349 726 B1	1/1990	(EP) .
0 363 567 B1	4/1990	(EP) .
8911349	4/1990	(EP) .
0 349 726	10/1990	(EP) .

Primary Examiner—Glenn A. Auve
(74) *Attorney, Agent, or Firm*—Ronald O. Neerings; Frederick J. Telecky, Jr.

(57) **ABSTRACT**

A real-time power conservation apparatus and method for portable computers employs a monitor to determine whether a CPU may rest based upon a real-time sampling of the CPU activity level and to activate a hardware selector to carry out the monitor's determination. If the monitor determines the CPU may rest, the hardware selector reduces CPU clock time; if the CPU is to be active, the hardware selector returns the CPU to its previous high speed clock level. Switching back into full operation from its rest state occurs without a user having to request it and without any delay in the operation of the computer while waiting for the computer to return to a "ready" state. Furthermore, the monitor adjusts the performance level of the computer to manage power conservation in response to the real-time sampling of CPU activity. Such adjustments are accomplished within the CPU cycles and do not affect the user's perception of performance and do not affect any system application software executing on the computer.

31 Claims, 6 Drawing Sheets



U.S. PATENT DOCUMENTS

4,686,386	8/1987	Tadao .
4,698,748	10/1987	Juzswik et al. .
4,748,559	5/1988	Smith et al. .
4,758,945	7/1988	Remedi .
4,780,843	10/1988	Tietjen .
4,814,591	3/1989	Nara et al. .
4,819,164	4/1989	Branson .
4,821,229	4/1989	Jauregui .
4,823,292	4/1989	Hillion .
4,823,309	4/1989	Kusaka et al. .
4,851,987	7/1989	Day .
4,870,570	9/1989	Satoh et al. .
4,893,271	1/1990	Davis et al. .
4,980,836	12/1990	Carter et al. .
5,025,387	6/1991	Frane .
5,083,266	1/1992	Watanabe .
5,086,387	2/1992	Arroyo et al. .
5,129,091	7/1992	Yorimoto et al. .
5,142,684	8/1992	Perry et al. .
5,167,024	11/1992	Smith .
5,179,693	1/1993	Kitamura et al. .
5,201,059	4/1993	Nguyen .
5,218,704	6/1993	Watts, Jr. et al. .
5,560,024	9/1996	Harper et al. .
5,930,516 *	7/1999	Watts, Jr. et al. 713/322

* cited by examiner

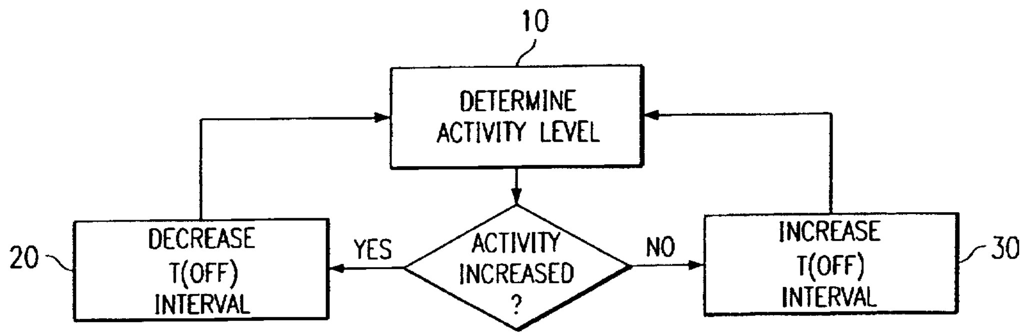


FIG. 1

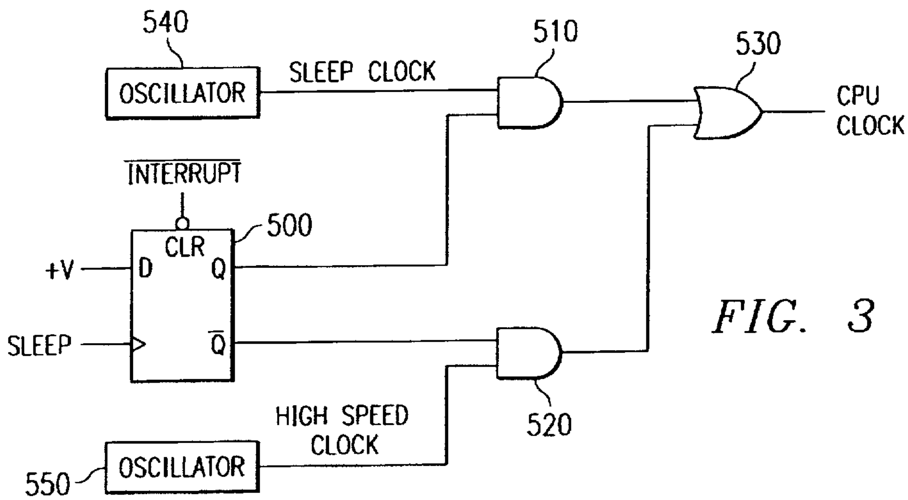


FIG. 3

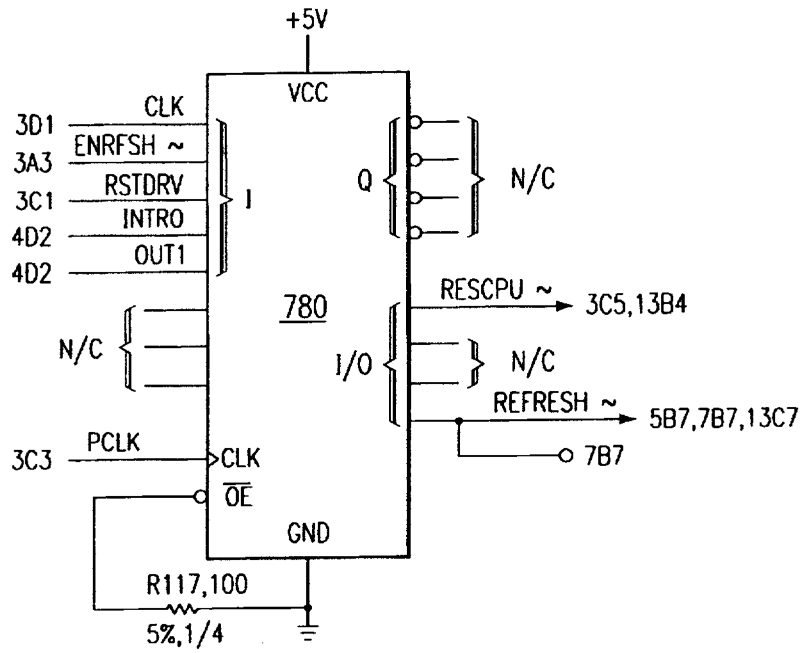
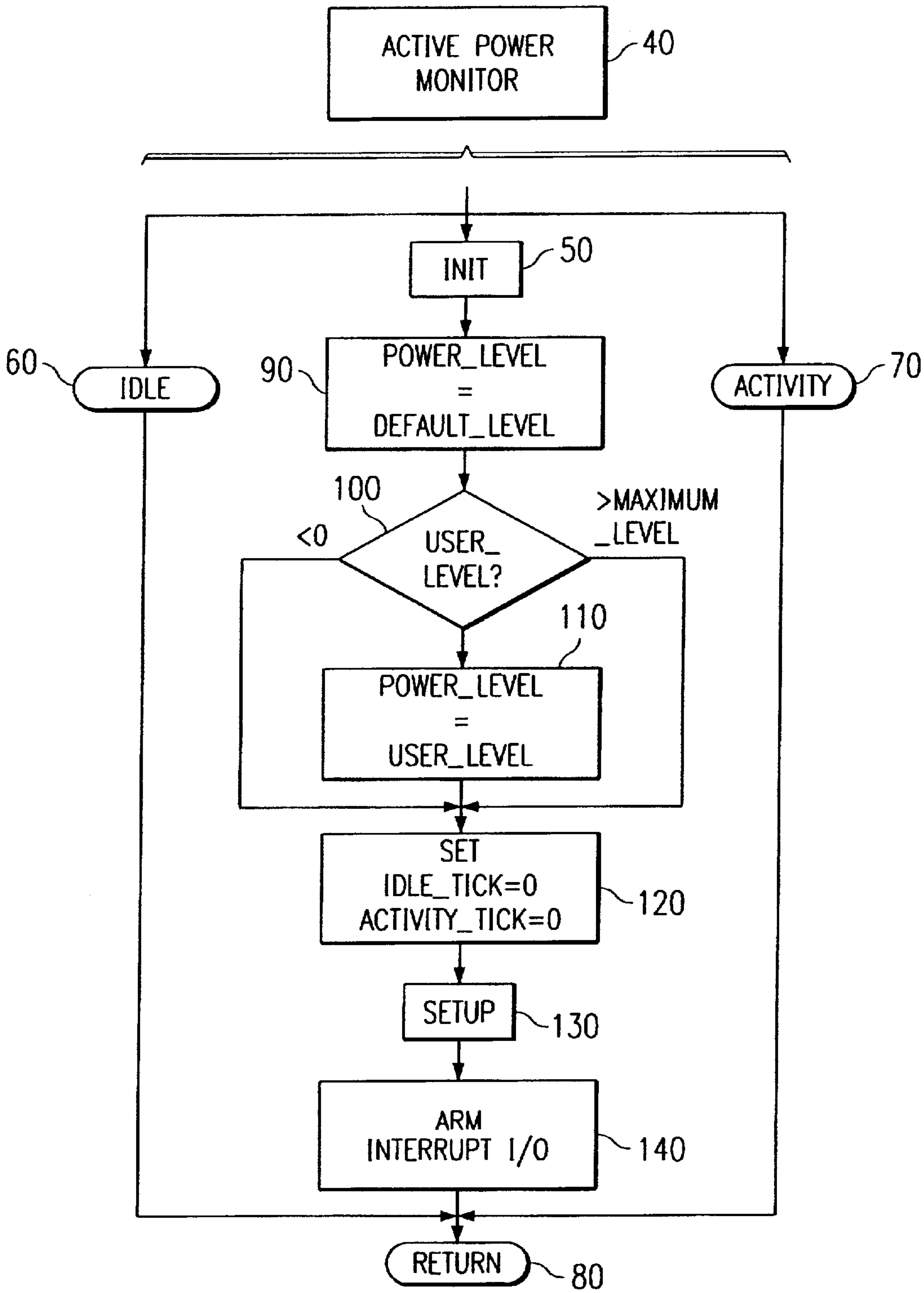


FIG. 5



ACTIVE POWER MONITOR
40
INIT
50
60
IDLE
90
POWER_LEVEL = DEFAULT_LEVEL
ACTIVITY
70
100
USER_ LEVEL?
<0
>MAXIMUM _LEVEL
110
POWER_LEVEL = USER_LEVEL
SET IDLE_TICK=0 ACTIVITY_TICK=0
120
SETUP
130
ARM INTERRUPT I/O
140
RETURN
80

FIG. 2a

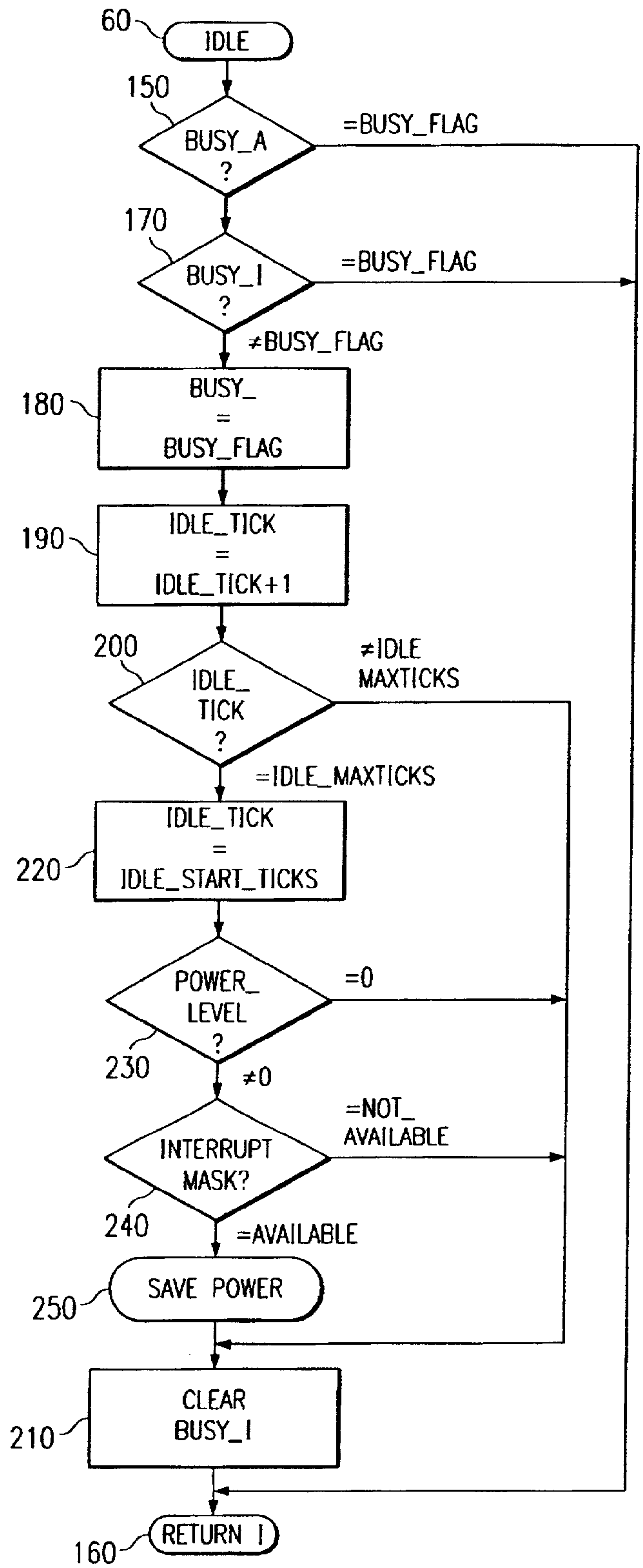

60
IDLE
150
BUSY_A ?
=BUSY_FLAG
170
BUSY_I ?
=BUSY_FLAG
≠BUSY_FLAG
180
BUSY_ = BUSY_FLAG
190
IDLE_TICK = IDLE_TICK+1
200
IDLE_ TICK ?
≠IDLE MAXTICKS
=IDLE_MAXTICKS
220
IDLE_TICK = IDLE_START_TICKS
POWER_ LEVEL ?
=0
230
≠0
INTERRUPT MASK?
=NOT_ AVAILABLE
240
=AVAILABLE
250
SAVE POWER
210
CLEAR BUSY_I
160
RETURN I

FIG. 2b

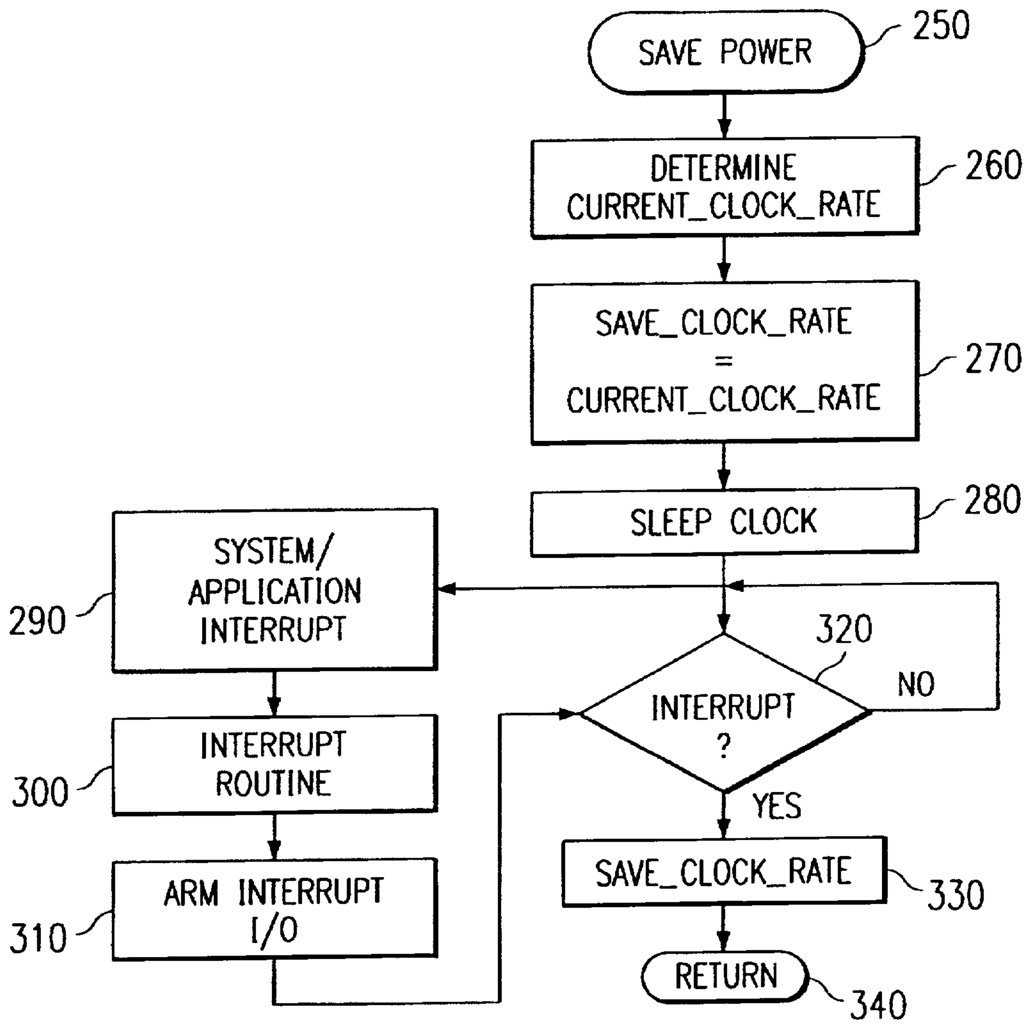
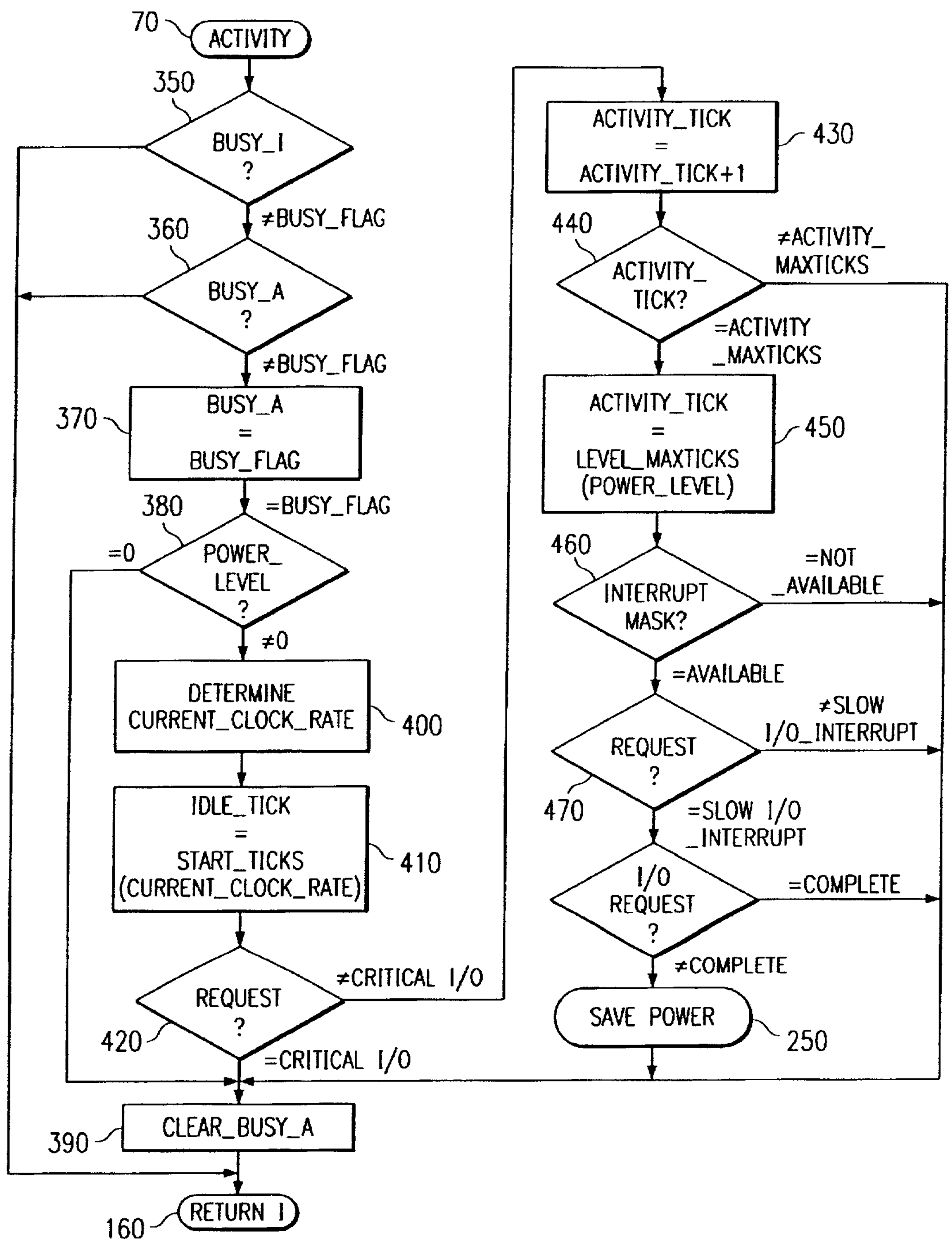

SAVE POWER
250
DETERMINE CURRENT_CLOCK_RATE
260
SAVE_CLOCK_RATE = CURRENT_CLOCK_RATE
270
SLEEP CLOCK
280
SYSTEM/ APPLICATION INTERRUPT
290
INTERRUPT ROUTINE
300
ARM INTERRUPT I/O
310
INTERRUPT ?
320
NO
YES
SAVE_CLOCK_RATE
330
RETURN
340

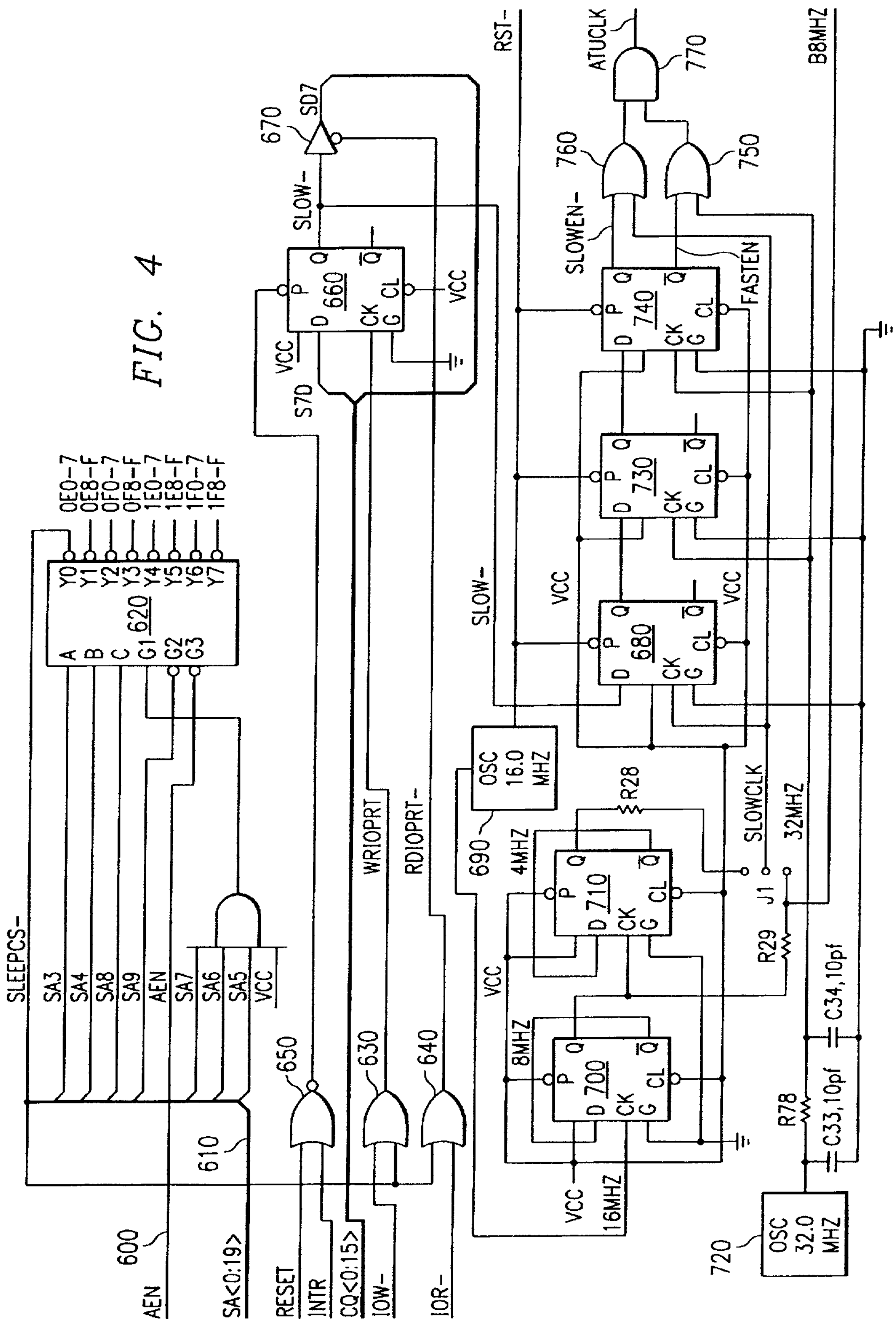
FIG. 2c



70
ACTIVITY
350
BUSY_I ?
≠BUSY_FLAG
360
BUSY_A ?
≠BUSY_FLAG
370
BUSY_A = BUSY_FLAG
380
=BUSY_FLAG
=0
POWER_ LEVEL ?
≠0
DETERMINE CURRENT_CLOCK_RATE
400
IDLE_TICK = START_TICKS (CURRENT_CLOCK_RATE)
410
REQUEST ?
≠CRITICAL I/O
420
=CRITICAL I/O
CLEAR_BUSY_A
390
RETURN I
160
ACTIVITY_TICK = ACTIVITY_TICK+1
430
440
ACTIVITY_ TICK?
≠ACTIVITY_ MAXTICKS
=ACTIVITY _MAXTICKS
ACTIVITY_TICK = LEVEL_MAXTICKS (POWER_LEVEL)
450
460
INTERRUPT MASK?
=NOT _AVAILABLE
=AVAILABLE
REQUEST ?
≠SLOW I/O_INTERRUPT
470
=SLOW I/O _INTERRUPT
I/O REQUEST ?
=COMPLETE
≠COMPLETE
SAVE POWER
250

FIG. 2d

FIG. 4


AEN
SA<0:19>
RESET
INTR
CQ<0:15>
IOW-
IOR-
600
610
650
630
640
SLEEPCS-
SA3
SA4
SA8
SA9
AEN
SA7
SA6
SA5
VCC
620
0E0-7
0E8-F
0F0-7
0F8-F
1E0-7
1E8-F
1F0-7
1F8-F
WRIOPRT
RDIOPRT-
660
670
SD7
S7D
SLOW-
RST-
ATUCLK
770
760
750
SLOWEN-
FASTEN
740
730
680
B8MHZ
OSC 16.0 MHZ
690
710
700
R28
SLOWCLK
32MHZ
J1
R29
4MHZ
8MHZ
16MHZ
R78
C33,10pf
C34,10pf
OSC 32.0 MHZ
720
VCC

REAL-TIME POWER CONSERVATION FOR ELECTRONIC DEVICE HAVING A PROCESSOR

This application is a Continuation of application Ser. No. 08/023,831 filed Apr. 12, 1993 U.S. Pat. No. 6,006,336, which is a Continuation of application Ser. No. 07/429,270 filed Oct. 30, 1989, now U.S. Pat. No. 5,218,704.

BACKGROUND OF THE INVENTION

1. Field of the Invention

This invention relates to real-time computer power conservation, and more particularly to an apparatus and method for reduction of central processing unit (CPU) clock time based on the real-time activity level within the CPU of a portable computer.

2. Description of the Related Art

During the development stages of personal computers, the transportable or portable computer has become very popular. Such portable computer uses a large power supply and really represents a small desktop personal computer. Portable computers are smaller and lighter than a desktop personal computer and allow a user to employ the same software that can be used on a desktop computer.

The first generation "portable" computers only operated from an A/C wall power. As personal computer development continued, battery-powered computers were designed. Furthermore, real portability became possible with the development of now display technology, better disk storage, and lighter components.

However, the software developed was desiged to run on desk top personal computers, with all the features of desktop computers, without regard to battery-powered portable computers that only had limited amounts of power available for short periods of time. No special considerations were made by the software, operating system (MS-DOS), Basic Input/Output System (BIOS), or the third party application software to conserve power usage for these portable computers.

As more and more highly functional software packages were developed, desk top computer users experienced increased performance from the introductions of higher computational CPUs, increased memory, and faster high performance disk drives.

Unfortunately, portable computers continued to run only on A/C power or with large and heavy batteries. In trying to keep up with the performance requirements of the desk top computers, and the new software, expensive components were used to cut the power requirements. Even so, the heavy batteries still did not run very long. This meant users of portable computers had to settle for A/C operation or very short battery operation to have the performance that was expected from the third party software.

Portable computer designers stepped the performance down to 8088- and 8086-type processors to reduce the power consumption. The supporting circuits and CPU took less power to run and therefore, lighter batteries could be used. Unfortunately, the new software requiring 80286-type instructions, that did not exist in the older slower 8088/8086 CPUs, did not run.

In an attempt to design a portable computer that could conserve power, thereby yielding longer battery operation, smaller units, and less weight, some portable computer designers proceeded to reduce power consumption of a portable computer while a user is not using the computer. For example, designers obtain a reduction in power usage by

slowing or stopping the disk drive after some predetermined period of inactivity; if the disk drive is not being used, the disk drive is turned off, or simply placed into a standby mode. When the user is ready to use the disk, the operator must wait until the disk drive spins up and the computer system is ready again for full performance before the operator may proceed with the operation.

Other portable computer designers conserve power by turning the computer display off when the keyboard is not being used. However, in normal operation the computer is using full power. In other words, power conservation by this method is practical only when the user is not using the components of the system. It is very likely, however, that the user will turn the computer off when not in use.

Nevertheless, substantial power conservation while the operator is using the computer for meaningful work is needed. When the operator uses the computer, full operation of all components is required. During the intervals while the operator is not using the computer, however, the computer could be turned off or slowed down to conserve power consumption. It is critical to maintaining performance to determine when to slow the computer down or turn it off-without disrupting the user's work, upsetting the third party software, or confusing the operating system, until operation is needed.

Furthermore, although an user can wait for the disk to spin up as described above, application software packages cannot wait for the CPU to "spin up" and get ready. The CPU must be ready when the application program needs to compute. Switching to full operation must be completed quickly and without the application program being affected. This immediate transition must be transparent to the user as well as to the application currently active. Delays cause user operational problems in response time and software compatability, as well as general failure by the computer to accurately execute a required program.

Other attempts at power conservation for portable computers include providing a "Shut Down" or "Standby Mode" of operation. The problem, again, is that the computer is not usable by the operator during this period. The operator could just as well turned off the power switch of the unit to save power. This type of power conservation only allows the portable computer to "shut down" and thereby save power if the operator forgets to turn off the power switch, or walks away from the computer for the programmed length of time. The advantage of this type of power conservation over just turning the power switch off/on is a much quicker return to full operation. However, this method of power conservation is still not real-time, intelligent power conservation while the computer is on and processing data which does not disturb the operating system, BIOS, and any third party application programs currently running on the computer.

Attempts to meet this need have been made by VLSI vendors in providing circuits that either turn off the clocks to the CPU when the user is not typing on the keyboard or wakes up the computer on demand when a keystroke occurred. Either of these approaches reduce power but the computer is dead (unusable) during this period. Background operations such as updating the system clock, communications, print spooling, and other like operations cannot be performed. Some existing portable computers employ these circuits. After a programmed period of no activity, the computer turns itself off. The operator must turn the machine on again but does not have to reboot the operating system and application program. The advantage of this circuitry is, like the existing "shut down" operations, a

quick return to full operation without restarting the computer. Nevertheless, this method only reduces power consumption when the user walks away from the machine and does not actually extend the operational life of the battery charge.

SUMMARY OF THE INVENTION

In view of the above problems associated with the related art, it is an object of the present invention to provide an apparatus and method for real-time conservation of power for computer systems without any real-time performance degradation, such conservation of power remaining transparent to the user.

Another object of the present invention is to provide an apparatus and method for predicting the activity level within a computer system and using the prediction for automatic power conservation.

Yet another object of the present invention is to provide an apparatus and method which allows user modification of automatic activity level predictions and using the modified predictions for automatic power conservation.

A further object of the present invention is to provide an apparatus and method for real-time reduction and restoration of clock speeds thereby returning the CPU to full processing rate from a period of inactivity which is transparent to software programs.

These objects are accomplished in a preferred embodiment of the present invention by an apparatus and method which determine whether a CPU may rest based upon the CPU activity level and activates a hardware selector based upon that determination. If the CPU may rest, or sleep, the hardware selector applies oscillations at a sleep clock level; if the CPU is to be active, the hardware selector applies oscillations at a high speed clock level.

The present invention examines the state of CPU activity, as well as the activity of both the operator and any application software program currently active. This sampling of activity is performed real-time, adjusting the performance level of the computer to manage power conservation and computer power. These adjustments are accomplished within the CPU cycles and do not affect the user's perception of performance.

Thus, when the operator for the third party software of the operating system/BIOS is not using the computer, the present invention will effect a quick turn off or slow down of the CPU until needed, thereby reducing the power consumption, and will promptly restore full CPU operation when needed without affecting perceived performance. This switching back into full operation from the "slow down" mode occurs without the user having to request it and without any delay in the operation of the computer while waiting for the computer to return to a "ready" state.

These and other features and advantages of the invention will be apparent to those skilled in the art from the following detailed description of a preferred embodiment, taken together with the accompanying drawings, in which:

DESCRIPTION OF THE DRAWINGS

FIG. 1 is a flowchart depicting the self-tuning aspect of a preferred embodiment of the present invention;

FIGS. *2a–2d* are flowcharts depicting the active power conservation monitor employed by the present invention;

FIG. 3 is a simplified schematic diagram representing the active power conservation associated hardware employed by the present invention;

FIG. 4 is a schematic of the sleep hardware for one embodiment of the present invention; and

FIG. 5 is a schematic of the sleep hardware for another embodiment of the present invention.

DETAILED DESCRIPTION OF A PREFERRED EMBODIMENT

If the period of computer activity in any given system is examined, the CPU and associated components have a utilization percentage. If the user is inputing data from the keyboard, the time between keystrokes is very long in terms of CPU cycles. Many things can be accomplished by the computer during this time, such as printing a report. Even during the printing of a report, time is still available for additional operations such as background updating of a clock/calendar display. Even so, there is almost always spare time when the CPU is not being used. If the computer is turned off or slowed down during this spare time, then power consumption is obtained real-time. Such real-time power conservation extends battery operation life.

According to the preferred embodiment of the present invention, to conserve power under MS-DOS, as well as other operating systems such as OS/2, XENIX, and those for Apple computers, requires a combination of hardware and software. It should be noted that because the present invention will work in any system, while the implementation may vary slightly on a system-by-system basis, the scope of the present invention should therefore not be limited to computer systems operating under MS/DOS.

Slowing down or stopping the computer system components according to the preferred embodiment of the present invention, reduces power consumption, although the amount of power saved may vary. Therefore, according to the present invention, stopping the clock (where possible as some CPUs cannot have their clocks stopped) reduces the power consumption more than just slowing the clock.

In general, the number of operations (or instructions) per second may be considered to be roughly proportional to the processor clock:

$$\text{instructions/second}=\text{instructions/cycle}*\text{cycles/second}$$

Assuming for simplicity that the same instruction is repeatedly executed so that instructions/second is constant, the relationship can be expressed as follows:

$$Fq=K_1*Clk$$

where Fq is instructions/second, K_1 is constant equal to the instructions/cycle, and Clk equals cycles/second. Thus, roughly speaking, the rate of execution increases with the frequency of the CPU clock.

The amount of power being used at any given moment is also related to the frequency of the CPU clock and therefore to the rate of execution. In general this relationship can be expressed as follows:

$$P=K_2+(K_3*Clk)$$

where P is power in watts, K_2 is a constant in watts, K_3 is a constant and expresses the number of watt-seconds/cycle, and Clk equals the cycles/second of the CPU clock. Thus it can also be said that the amount of power being consumed at any given time increases as the CPU clock frequency increases.

Assume that a given time period T is divided into N intervals such that the power P is constant during each

interval. Then the amount of energy E expended during T is given by:

$$E=P(1)\text{delta}T_1+P(2)\text{delta}T_2 \ldots +P(N)\text{delta}T_N$$

Further assume that the CPU clock "Clk" has only two states, either "ON" or "OFF". For the purposes of this discussion, the "ON" state represents the CPU clock at its maximum frequency, while the "OFF" state represents the minimum clock rate at which the CPU can operate (this may be zero for CPUs that can have their clocks stopped). For the condition in which the CPU clock is always "ON", each P(i) in the previous equation is equal and the total energy is:

$$\begin{aligned} E(\max) &= P(\text{on})*(\text{delta } T_1 + \text{delta } T_2 \ldots + \text{delta } T_N) \\ &= P(\text{on})*T \end{aligned}$$

This represents the maximum power consumption of the computer in which no power conservation measures are being used. If the CPU clock is "off" during a portion of the intervals, then there are two power levels possible for each interval. The P(on) represents the power being consumed when the clock in in its "ON" state, while P(off) represents the power being used when the clock is "OFF". If all of the time intervals in which the clock is "ON" is summed into the quantity "T(on)" and the "OFF" intervals are summed into "T(off)", then it follows:

$$T=T(\text{on})+T(\text{off})$$

Now the energy being used during period T can be written:

$$E=[P(\text{on})*T(\text{on})]+[P(\text{off})*T(\text{off})]$$

Under these conditions, the total energy consumed may be reduced by increasing the time intervals T(off). Thus, by controlling the periods of time the clock is in its "OFF" state, the amount of energy being used may be reduced. If the T(off) period is divided into a large number of intervals during the period T, then as the width of each interval goes to zero, energy consumption is at a maximum. Conversely, as the width of the T(off) intervals increase, the energy consumed decreases.

If the "OFF" intervals are arranged to coincide with periods during which the CPU is normally inactive, then the user cannot perceive any reduction in performance and overall energy consumption is reduced from the E(max) state. In order to align the T(off) intervals with periods of CPU inactivity, the CPU activity level is used to determine the width of the T(off) intervals in a closed loop. FIG. **1** depicts such a closed loop. The activity level of the CPU is determined at Step **10**. If this level is an increase over an immediately previous determination, the present invention decreases the T(off) interval (Step **20**) and returns to determine the activity level of the CPU again. If, on the other hand, this activity level is a decrease over an immediately previous determination, the present invention increases the T(off) interval (Step **30**) and proceeds to again determine the activity level of the CPU. Thus the T(off) intervals are constantly being adjusted to match the system activity level.

In any operating system, two key logic points exist: an IDLE, or "do nothing", loop within the operating system and an operating system request channel, usually available for services needed by the application software. By placing logic inline with these logic points, the type of activity request made by an application software can be evaluated, power conservation can be activated and slice periods determined. A slice period is the number of T(on) vs. T(off) intervals over time, computed by the activity level. An assumption may be made to determine CPU activity level: Software programs that need service usually need additional services and the period of time between service requests can be used to determine the activity level of any application software running on the computer and to provide slice counts for power conservation according to the present invention.

Once the CPU is interrupted during a power conservation slice (T(off)), the CPU will save the interrupted routine's state prior to vectoring to the interrupt software. Of course, since the power conservation software was operating during this slice, control will be returned to the active power conservation loop (monitor **40**) which simply monitors the CPU's clock to determine an exit condition for the power conservation mode, thereby exiting from T(off) to T(on) state. The interval of the next power conservation state is adjusted by the activity level monitored, as discussed above in connection with FIG. **1**. Some implementations can create an automatic exit from T(off) by the hardware logic, thereby forcing the power conservation loop to be exited automatically and executing an interval T(on).

More specifically, looking now at FIGS. **2***a*–**2***d*, which depict the active power conservation monitor **40** of the present invention. The CPU installs monitor **40** either via a program stored in the CPU ROM or loads it from an external device storing the program in RAM. Once the CPU has loaded monitor **40**, it continues to INIT **50** for system interrupt initialization, user configurational setup, and system/application specific initialization. IDLE branch **60** (more specifically set out in FIG. **2***b*) is executed by a hardware or software interrupt for an IDLE or "do nothing" function. This type of interrupt is caused by the CPU entering either an IDLE or a "do nothing" loop (i.e., planned inactivity). The ACTIVITY branch **70** of the flowchart, more fully described below in relation to FIG. **2***d*, is executed by a software or hardware interrupt due to an operating system or I/O service request, by an application program or internal operating system function. An I/O service request made by a program may, for example, be a disk I/O, read, print, load, etc. Regardless of the branch selected, control is eventually returned to the CPU operating system at RETURN **80**. The INIT branch **50** of this flowchart, shown in FIG. **2***a*, is executed only once if it is loaded via program into ROM or is executed every time during power up if it is loaded from an external device and stored in the RAM. Once this branch of active power monitor **40** has been fully executed, whenever control is yielded from the operating system to the power conservation mode, either IDLE **60** or ACTIVITY **70** branches are selected depending on the type of CPU activity: IDLE branch **60** for power conservation during planned inactivity and ACTIVITY branch **70** for power conservation during CPU activity.

Looking more closely at INIT branch **50**, after all system interrupt and variables are initialized, the routine continues at Step **90** to set the Power_level equal to DEFAULT_LEVEL. In operating systems where the user has input control for the Power_level, the program at Step **100** checks to see if a User_level has been selected. If the User level is less than zero or greater than the MAXIMUM_LEVEL, the system uses the DEFAULT_LEVEL. Otherwise, it continues onto Step **110** where it modifies the Power_level to equal the User_level.

According to the preferred embodiment of the present invention, the system at Step **120** sets the variable Idle_tick to zero and the variable Activity_tick to zero. Under an

MS/DOS implementation, Idle_tick refers to the number of interrupts found in a "do nothing" loop. Activity_tick refers to the number of interrupts caused by an activity interrupt which in turn determines the CPU activity level. Tick count represents a delta time for the next interrupt. Idle_tick is a constant delta time from one tick to another (interrupt) unless overwritten by a software interrupt. A software interrupt may reprogram delta time between interrupts.

After setting the variables to zero, the routine continues on to Setup **130** at which time any application specific configuration fine-tuning is handled in terms of system-specific details and the system is initialized. Next the routine arms the interrupt I/O (Step **140**) with instructions to the hardware indicating the hardware can take control at the next interrupt. INIT branch **50** then exits to the operating system, or whatever called the active power monitor originally, at RETURN **80**.

Consider now IDLE branch **60** of active power monitor **40**, more fully described at FIG. 2*b*. In response to a planned inactivity of the CPU, monitor **40** (not specifically shown in this Figure) checks to see if entry into IDLE branch **60** is permitted by first determining whether the activity interrupt is currently busy. If Busy_A equals BUSY_FLAG (Step **150**), which is a reentry flag, the CPU is busy and cannot now be put to sleep. Therefore, monitor **40** immediately proceeds to RETURN I **160** and exits the routine. RETURN I **160** is an indirect vector to the previous operating system IDLE vector interrupt for normal processing stored before entering monitor **40**. (I.e., this causes an interrupt return to the last chained vector.)

If the Busy_A interrupt flag is not busy, then monitor **40** checks to see if the Busy_Idle interrupt flag, Busy_I, equals BUSY_FLAG (Step **170**). If so, this indicates the system is already in IDLE branch **60** of monitor **40** and therefore the system should not interrupt itself. If Busy_I=BUSY_FLAG, the system exits the routine at RETURN I indirect vector **160**.

If, however, neither the Busy_A reentry flag or the Busy_I reentry flag have been set, the routine sets the Busy_I flag at Step **180** for reentry protection (Busy_I=BUSY_FLAG). At Step **190** Idle_tick is incremented by one. Idle_tick is the number of T(on) before a T(off) interval and is determined from IDLE interrupts, setup interrupts and from CPU activity level. Idle_tick increments by one to allow for smoothing of events, thereby letting a critical I/O activity control smoothing.

At Step **200** monitor **40** checks to see if Idle_tick equals IDLE_MAXTICKS. IDLE_MAXTICKS is one of the constants initialized in Setup **130** of INIT branch **50**, remains constant for a system, and is responsible for self-tuning of the activity level. If Idle_tick does not equal IDLE_MAXTICKS, the Busy_I flag is cleared at Step **210** and exits the loop proceeding to the RETURN I indirect vector **160**. If, however, Idle_tick equals IDLE_MAXTCKS, Idle_tick is set equal to IDLE_START_TICKS (Step **220**). IDLE_START_TICKS is a constant which may or may not be zero (depending on whether the particular CPU can have its clock stopped). This step determines the self-tuning of how often the rest of the sleep functions may be performed. By setting IDLE_START_TICKS equal to IDLE_MAXTICKS minus one, a continuous T(off) interval is achieved. At Step **230**, the Power_level is checked. If it is equal to zero, the monitor clears the Busy_I flag (Step **210**), exits the routine at RETURN I **160**, and returns control to the operating system so it may continue what it was originally doing before it entered active power monitor **40**.

If, however, the Power_level does not equal zero at Step **240**, the routine determines whether an interrupt mask is in place. An interrupt mask is set by the system/application software, and determines whether interrupts are available to monitor **40**. If interrupts are NOT_AVAILABLE, the Busy_I reentry flag is cleared and control is returned to the operating system to continue what it was doing before it entered monitor **40**. Operating systems, as well as application software, can set T(on) interval to yield a continuous T(on) state by setting the interrupt mask equal to NOT_AVAILABLE.

Assuming an interrupt is AVAILABLE, monitor **40** proceeds to the SAVE POWER subroutine **250** which is fully executed during one T(off) period established by the hardware state. (For example, in the preferred embodiment of the present invention, the longest possible interval could be 18 ms, which is the longest time between two ticks or interrupts from the real-time clock.) During the SAVE POWER subroutine **250**, the CPU clock is stepped down to a sleep clock level.

Once a critical I/O operation forces the T(on) intervals, the IDLE branch **60** interrupt tends to remain ready for additional critical I/O requests. As the CPU becomes busy with critical I/O, less T(off) intervals are available. Conversely, as critical I/O requests decrease, and the time intervals between them increase, more T(off) intervals are available. IDLE branch **60** is a self-tuning system based on feedback from activity interrupts and tends to provide more T(off) intervals as the activity level slows. As soon as monitor **40** has completed SAVE POWER subroutine **250**, shown in FIG. 2*c* and more fully described below, the Busy_I reentry flag is cleared (Step **210**) and control is returned at RETURN I **160** to whatever operating system originally requested monitor **40**.

Consider now FIG. 2*c*, which is a flowchart depicting the SAVE POWER subroutine **250**. Monitor **40** determines what the I/O hardware high speed clock is at Step **260**. It sets the CURRENT_CLOCK_RATE equal to the relevant high speed clock and saves this value to be used for CPUs with multiple level high speed clocks. Thus, if a particular CPU has 12 MHz and 6 MHz high speed clocks, monitor **40** must determine which high speed clock the CPU is at before monitor **40** reduces power so it may reestablish the CPU at the proper high speed clock when the CPU awakens. At Step **270**, the Save_clock_rate is set equal to the CURRENT_CLOCK_RATE determined. Save_clock_rate **270** is not used when there is only one high speed clock for the CPU. Monitor **40** now continues to SLEEPCLOCK **280**, where a pulse is sent. to the hardware selector (shown in FIG. **3**) to put the CPU clock to sleep (i.e., lower or stop its clock frequency). The I/O port hardware sleep clock is at much lower oscillations than the CPU clock normally employed.

At this point either of two events can happen. A system/application interrupt may occur or a real-time clock interrupt may occur. If a system/application interrupt **290** occurs, monitor **40** proceeds to interrupt routine **300**, processing the interrupt as soon as possible, arming interrupt I/O at Step **310**, and returning to determine whether there has been an interrupt (Step **320**). Since in this case there has been an interrupt, the Save_clock_rate is used (Step **330**) to determine which high speed clock to return the CPU to and SAVE POWER subroutine **250** is exited at RETURN **340**. If, however, a system/application interrupt is not received, the SAVE POWER subroutine **250** will continue to wait until a real-time clock interrupt has occurred (Step **320**). Once such an interrupt has occurred, SAVE POWER subroutine **250** reestablishes the CPU at the stored Save-clock-rate. If the sleep clock rate was not stopped, in other words, the sleep clock rate was not zero, control is passed at a slow clock and

SAVE POWER subroutine **250** will execute interrupt loop **320** several times. If however, control is passed when the sleep clock rate was zero, in other words, there was no clock, the SAVE POWER subroutine **250** will execute interrupt loop **320** once before returning the CPU clock to the Save_clock_rate **330** and exiting (Step (**340**).

Consider now FIG. 2*d* which is a flowchart showing ACTIVITY branch **70** triggered by an application/system activity request via an operating system service request interrupt. ACTIVITY branch **70** begins with reentry protection. Monitor **40** determines at Step **350** whether Busy_I has been set to BUSY_FLAG. If it has, this means the system is already in IDLE branch **60** and cannot be interrupted. If Busy_I=BUSY_FLAG, monitor **40** exits to RETURN I **160**, which is an indirect vector to an old activity vector interrupt for normal processing, via an interrupt vector after the operating system performs the requested service.

If however, the Busy_I flag does not equal BUSY_FLAG, which means IDLE branch **60** is not being accessed, monitor **40** determines at Step **360** if the BUSY_A flag has been set equal to BUSY_FLAG. If so, control will be returned to the system at this point because ACTIVITY branch **70** is already being used and cannot be interrupted. If the Busy_A flag has not been set, in other words, Busy_A does not equal BUSY_FLAG, monitor **40** sets Busy_A equal to BUSY_FLAG at Step **370** so as not to be interrupted during execution of ACTIVITY branch **70**. At Step **380** the Power_level is determined. If Power_level equals zero, monitor **40** exits ACTIVITY branch **70** after clearing the Busy_A reentry flag (Step **390**). If however, the Power_level does not equal zero, the CURRENT_CLOCK_RATE of the I/O hardware is next determined. As was true with Step **270** of FIG. 2C, Step **400** of FIG. 2*d* uses the CURRENT_CLOCK_RATE if there are multiple level high speed clocks for a given CPU. Otherwise, CURRENT_CLOCK_RATE always equals the CPU high speed clock. After the CURRENT_CLOCK_RATE is determined (Step **400**), at Step **410** Idle_tick is set equal to the constant START_TICKS established for the previously determined CURRENT_CLOCK_RATE. T(off) intervals are established based on the current high speed clock that is active.

Monitor **40** next determines that a request has been made. A request is an input by the application software running on the computer, for a particular type of service needed. At Step **420**, monitor **40** determines whether the request is a CRITICAL I/O. If the request is a CRITICAL I/O, it will continuously force T(on) to lengthen until the T(on) is greater than the T(off), and monitor **40** will exit ACTIVITY branch **70** after clearing the Busy_A reentry flag (Step **390**). If, on the other hand, the request is not a CRITICAL I/O, then the Activity_tick is incremented by one at Step **430**. It is then determined at Step **440** whether the Activity_tick now equals ACTIVITY_MAXTICKS. Step **440** allows a smoothing from a CRITICAL I/O, and makes the system ready from another CRITICAL I/O during Activity_tick T(on) intervals. Assuming Activity tick does not equal ACTIVITY_MAXTICKS, ACTIVITY branch **70** is exited after clearing the Busy_A reentry flag (Step **390**). If, on the other hand, the Activity tick equals constant ACTIVITY_MAXTICKS, at Step **450** Activity_tick is set to the constant LEVEL_MAXTICKS established for the particular Power_level determined at Step **380**.

Now monitor **40** determines whether an interrupt mask exists (Step **460**). An interrupt mask is set by system/application software. Setting it to NOT_AVAILABLE creates a continuous T(on) state. If the interrupt mask equals NOT_AVAILABLE, there are no interrupts available at this time and monitor **40** exits ACTIVITY branch **70** after clearing the Busy_A reentry flag (Step **390**). If, however, an interrupt is AVAILABLE, monitor **40** determines at Step **470** whether the request identified at Step **420** was for a SLOW I/O_INTERRUPT. SLOW I/O requests may have a delay until the I/O device becomes "ready". During the "make ready" operation, a continuous T(off) interval may be set up and executed to conserve power. Thus, if the request is not a SLOW I/O_INTERRUPT, ACTIVITY branch **70** is exited after clearing the Busy_A reentry flag (Step **390**). If, however, the request is a SLOW I/O_INTERRUPT, and time yet exists before the I/O device becomes "ready", monitor **40** then determines at Step **480** whether the I/O request is COMPLETE (i.e., is I/O device ready?). If the I/O device is not ready, monitor **40** forces T(off) to lengthen, thereby forcing the CPU to wait, or sleep, until the SLOW I/O device is ready. At this point it has time to save power and ACTIVITY branch **70** enters SAVE POWER subroutine **250** previously described in connection with to FIG. 2C. If, however, the I/O request is COMPLETE, control is returned to the operating system subsequently to monitor **40** exiting ACTIVITY branch **70** after clearing Busy_A reentry flag (Step **390**).

Self-tuning is inherent within the control system of continuous feedback loops. The software of the present invention can detect when CPU activity is low and therefore when the power conservation aspect of the present invention may be activated. Once the power conservation monitor is activated, a prompt return to full speed CPU clock operation within the interval is achieved so as to not degrade the performance of the computer. To achieve this prompt return to full speed CPU clock operation, the preferred embodiment of the present invention employs some associated hardware.

Looking now at FIG. **3** which shows a simplified schematic diagram representing the associated hardware employed by the present invention for active power conservation. When monitor **40** (not shown) determines the CPU is ready to sleep, it writes to an I/O port (not shown) which causes a pulse on the SLEEP line. The rising edge of this pulse on the SLEEP line causes flip flop **500** to clock a high to Q and a low to Q−. This causes the AND/OR logic (AND gates **510**, **520**; OR gate **530**) to select the pulses travelling the SLEEP CLOCK line from SLEEP CLOCK oscillator **540** to be sent to and used by the CPU CLOCK. SLEEP CLOCK oscillator **540** is a slower clock than the CPU clock used during normal CPU activity. The high coming from the Q of flip flop **500** ANDed (**510**) with the pulses coming from SLEEP CLOCK oscillator **540** is ORed (**530**) with the result of the low on the Q− of flip flop **500** ANDed (**520**) with the pulse generated along the HIGH SPEED CLOCK line by the HIGH SPEED CLOCK oscillator **550** to yield the CPU CLOCK. When the I/O port designates SLEEP CLOCK, the CPU CLOCK is then equal to the SLEEP CLOCK oscillator **540** value. If, on the other hand, an interrupt occurs, an interrupt-value clears flip flop **500**, thereby forcing the AND/OR selector (comprising **510**, **520** and **530**) to choose the HIGH SPEED CLOCK value, and returns the CPU CLOCK value to the value coming from HIGH SPEED CLOCK oscillator **550**. Therefore, during any power conservation operation on the CPU, the detection of any interrupt within the system will restore the CPU operation at full clock rate prior to vectoring and processing the interrupt.

It should be noted that the associated hardware needed, external to each of the CPUs for any given system, may be different depending upon the operating system used,

whether the CPU can be stopped, etc. Nevertheless, the scope of the present invention should not be limited by possible system specific modifications needed to permit the present invention to actively conserve power in the numerous available portable computer systems. For example two actual implementations are shown in FIGS. 4 and 5, discussed below.

Many VSLI designs today allow for clock switching of the CPU speed. The logic to switch from a null clock or slow clock to a fast clock logic is the same as that which allows the user to change speeds by a keyboard command. The added logic of monitor **40** working with such switching logic, causes an immediate return to a fast clock upon detection of any interrupt. This simple logic is the key to the necessary hardware support to interrupt the CPU and thereby allow the processing of the interrupt at full speed.

The method to reduce power consumption under MS-DOS employs the MS-DOS IDLE loop trap to gain access to the "do nothing" loop. The IDLE loop provides special access to application software and operating system operations that are in a state of IDLE or low activity. Careful examination is required to determine the activity level at any given point within the system. Feedback loops are used from the interrupt 21H service request to determine the activity level. The prediction of activity level is determined by interrupt 21H requests, from which the present invention thereby sets the slice periods for "sleeping" (slowing down or stopping) the CPU. An additional feature allows the user to modify the slice depending on the activity level of interrupt 21H.

Looking now at FIG. **4**, which depicts a schematic of an actual sleep hardware implementation for a system such as the Intel 80386 (CPU cannot have its clock stopped). Address enable bus **600** and address bus **610** provide CPU input to demultiplexer **620**. The output of demultiplexer **620** is sent along SLEEPCS- and provided as input to OR gates **630,640**. The other inputs to OR gates **630,640** are the I/O write control line and the I/O read control line, respectively. The outputs of these gates, in addition to NOR gate **650**, are applied to D flip flop **660** to decode the port. "INTR" is the interrupt input from the I/O port (peripherals) into NOR gate **650**, which causes the logic hardware to switch back to the high speed clock. The output of flip flop **660** is then fed, along with the output from OR gate **630**, to tristate buffer **670** to enable it to read back what is on the port. All of the above-identified hardware is used by the read/write I/O port (peripherals) to select the power saving "Sleep" operation. The output "SLOW-" is equivalent to "SLEEP" in FIG. 2, and is inputted to flip flop **680**, discussed later.

The output of SLEEP CLOCK oscillator **690** is divided into two slower clocks by D flip flops **700,710**. In the particular implementation shown in FIG. **4**, 16 MHz sleep clock oscillator **690** is divided into 4 MHz and 8 MHz clocks. Jumper **J1** selects which clock is to be the "SLEEP CLOCK".

In this particular implementation, high speed clock oscillator **720** is a 32 MHz oscillator, although this particular speed is not a requirement of the present invention. The 32 MHz oscillator is put in series with a resistor (for the implementation shown, 33 ohms), which is in series with two parallel capacitors (10 pF). The result of such oscillations is tied to the clocks of D flip flops **730,740**.

D flip flops **680,730,740** are synchronizing flip flops; **680,730** were not shown in the simplified sleep hardware of FIG. 2. These flip flops are used to ensure the clock switch occurs only on clock edge. As can be seen in FIG. **4**, as with flip flop **500** of FIG. 2, the output of flip flop **740** either activates OR gate **750** or OR gate **760**, depending upon whether the CPU is to sleep ("FASTEN-") or awaken ("SLOWEN-").

OR gates **750,760** and AND gate **770** are the functional equivalents to the AND/OR selector of FIG. **2**. They are responsible for selecting either the "slowclk" (slow clock, also known as SLEEP CLOCK) or high speed clock (designated as 32 MHz on the incoming line). In this implementation, the Slow clock is either 4 MHz or 8 MHz, depending upon jumper J1, and the high speed clock is 32 MHz. The output of AND gate **770** (ATUCLK) establishes the rate of the CPU clock, and is the equivalent of CPU CLOCK of FIG. 2.

Consider now FIG. **5**, which depicts a schematic of another actual sleep hardware implementation for a system such as the Intel 80286 (CPU can have its clock stopped). The Western Digital FE3600 VLSI is used for the speed switching with a special external PAL 780 to control the interrupt gating which wakes up the CPU on any interrupt. The software power conservation according to the present invention monitors the interrupt acceptance, activating the next P(i)deltaT$_i$ interval after the interrupt.

Any interrupt request to the CPU will return the system to normal operation. An interrupt request ("INTRQ") to the CPU will cause the PAL to issue a Wake Up signal on the RESCPU line to the FE3001 (not shown) which in turn enables the CPU and the DMA clocks to bring the system back to its normal state. This is the equivalent of the "INTERRUPT-" of FIG. **2**. Interrupt Request is synchronized to avoid confusing the state machine so that Interrupt (INTDET) will only be detected while the cycle is active. The rising edge of RESCPU will wake up the FE 3001 which in turn releases the whole system from the Sleep Mode.

Implementation for the 386SX is different only in the external hardware and software power conservation loop. The software loop will set external hardware to switch to the high speed clock on interrupt prior to vectoring the interrupt. Once return is made to the power conservation software, the high speed clock cycle will be detected and the hardware will be reset for full clock operation.

Implementation for OS/2 uses the "do nothing" loop programmed as a THREAD running in background operation with low priority. Once the THREAD is activated, the CPU sleep, or low speed clock, operation will be activated until an interrupt occurs thereby placing the CPU back to the original clock rate.

Although interrupts have been employed to wake up the CPU in the preferred embodiment of the present invention, it should be realized that any periodic activity within the system, or applied to the system, could also be used for the same function.

While several implementations of the preferred embodiment of the invention has been shown and described, various modifications and alternate embodiments will occur to those skilled in the art. Accordingly, it is intended that the invention be limited only in terms of the appended claims.

We claim:

1. An apparatus, comprising:

a central processing unit (CPU) having a monitor for measuring the relative amount of idle time within said CPU; and

a clock manager coupled to said CPU, said clock manager selectively modifying a clock signal being sent to said CPU.

2. An apparatus, comprising:

a central processing unit (CPU) having a monitor for measuring the relative amount of idle time within said CPU; and

a clock manager coupled to said CPU, said clock manager selectively modifying a clock signal being sent to said CPU in response to usage of said CPU being below a preselected level.

3. An apparatus, comprising:

a central processing unit (CPU) having a monitor for measuring the relative amount of idle time within said CPU; and

a clock manager coupled to said CPU, said clock manager selectively modifying a clock signal being sent to said CPU to reduce the idle time in said CPU.

4. The apparatus of claim 3, wherein said monitor inhibits the modification of said clock signal while said CPU is processing critical I/O.

5. The apparatus of claim 3, wherein said CPU sends signals to the clock manager requesting the clock manager to demodify the clock signal being sent to the CPU in response to said monitor detecting a critical I/O request.

6. The apparatus of claim 3, wherein said monitor is self-tuning.

7. The apparatus of claim 6, wherein said monitor uses a control system of continuous feedback loops.

8. An apparatus, comprising:

a central processing unit (CPU) having a monitor for measuring the relative amount of idle time within said CPU; and

a clock manager coupled to said CPU, said clock manager selectively modifying a clock signal being sent to said CPU to minimize the relative amount of idle time in said CPU.

9. An apparatus, comprising:

a central processing unit (CPU) having a monitor for measuring the relative amount of activity time within said CPU; and

a clock manager coupled to said CPU, said clock manager selectively modifying a clock signal being sent to said CPU.

10. The apparatus of claim 9, wherein said monitor inhibits the modification of said clock signal while said CPU is processing critical I/O.

11. The apparatus of claim 9, wherein said CPU sends signals to the clock manager requesting the clock manager to demodify the clock signal being sent to the CPU in response to said monitor detecting a critical I/O request.

12. The apparatus of claim 9, wherein said monitor is self-tuning.

13. The apparatus of claim 12, wherein said monitor uses a control system of continuous feedback loops.

14. An apparatus, comprising:

a central processing unit (CPU) having a monitor for measuring the relative amount of activity time within said CPU; and

a clock manager coupled to said CPU, said clock manager selectively modifying a clock signal being sent to said CPU in response to usage of said CPU being below a preselected level.

15. An apparatus, comprising:

a central processing unit (CPU) having a monitor for measuring the relative amount of activity time within said CPU; and

a clock manager coupled to said CPU, said clock manager selectively modifying a clock signal being sent to said CPU to control the amount of activity time in said CPU.

16. An apparatus, comprising:

a central processing unit (CPU) having a monitor for measuring the relative amount of activity time within said CPU; and

a clock manager coupled to said CPU, said clock manager selectively modifying a clock signal being sent to said CPU to optimize the activity time within said CPU in response to usage of said CPU being below a preselected level.

17. An apparatus, comprising:

a central processing unit (CPU) having a monitor for measuring the relative amount of idle time and activity time within said CPU; and

a clock manager coupled to said CPU, said clock manager selectively modifying a clock signal being sent to said CPU.

18. An apparatus, comprising:

a central processing unit (CPU) having a monitor for measuring the relative amount of idle time and activity time within said CPU; and

a clock manager coupled to said CPU, said clock manager selectively modifying a clock signal being sent to said CPU in response to usage of said CPU being below a preselected level.

19. An apparatus, comprising:

a central processing unit (CPU) having a monitor for measuring the relative amount of idle time and activity time within said CPU; and

a clock manager coupled to said CPU, said clock manager selectively modifying a clock signal being sent to said CPU to control the amount of idle time and activity time in said CPU.

20. The apparatus of claim 19, wherein said monitor inhibits the modification of said clock signal while said CPU is processing critical I/O.

21. The apparatus of claim 19, wherein said CPU sends signals to the clock manager requesting the clock manager to demodify the clock signal being sent to the CPU in response to said monitor detecting a critical I/O request.

22. The apparatus of claim 19, wherein said monitor is self-tuning.

23. The apparatus of claim 22, wherein said monitor uses a control system of continuous feedback loops.

24. An apparatus, comprising:

a central processing unit (CPU) having a monitor for measuring the relative amount of idle time and activity time within said CPU; and

a clock manager coupled to said CPU, said clock manager selectively modifying a clock signal being sent to said CPU to control the amount of idle time and activity time in said CPU in response to a utilization percentage of said CPU being below a preselected level.

25. An apparatus, comprising:

a central processing unit (CPU) having a monitor for measuring the utilization of said CPU; and

a clock manager coupled to said CPU, said clock manager selectively modifying a clock signal being sent to said CPU to control a utilization percentage of said CPU.

26. An apparatus, comprising:

a central processing unit (CPU) coupled to a clock and having a monitor for measuring the relative amount of idle time and activity time within said CPU; and

a clock manager coupled to said CPU, said clock manager controlling periods of time said clock is in an OFF state, the length of said periods of time said clock is in an OFF state being appropriate to allow said CPU to operate at an efficient utilization percentage.

27. The apparatus of claim 26, wherein energy consumption in said CPU is at a maximum when the length of each period of time said clock is in an OFF state is at zero.

28. The apparatus of claim **26**, wherein energy consumption in said CPU decreases as the length of each period of time said clock is in an OFF state increases.

29. The apparatus of claim **26**, wherein said periods of time said clock is in an OFF state are constantly being adjusted to optimize said utilization percentage of said central processing unit.

30. The apparatus of claim **26**, wherein said OFF state represents the minimum clock rate at which said central processing unit can operate.

31. The apparatus of claim **26**, wherein said minimum clock rate may be zero for central processing units that can have their clocks stopped.

* * * * *

EXHIBIT 3

US006397340B2

(12) **United States Patent**
Watts, Jr. et al.

(10) **Patent No.: US 6,397,340 B2**
(45) **Date of Patent: May 28, 2002**

(54) **REAL-TIME POWER CONSERVATION FOR ELECTRONIC DEVICE HAVING A PROCESSOR**

(75) Inventors: **LaVaughn F. Watts, Jr.**, Austin; **Steven J. Wallace**, Waco, both of TX (US)

(73) Assignee: **Texas Instruments Incorporated**, Dallas, TX (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/756,838**

(22) Filed: **Jan. 9, 2001**

Related U.S. Application Data

(63) Continuation of application No. 09/392,205, filed on Sep. 8, 1999, now Pat. No. 6,173,409, which is a continuation of application No. 08/023,831, filed on Apr. 12, 1993, now Pat. No. 6,006,336, which is a continuation of application No. 07/429,270, filed on Oct. 30, 1989, now Pat. No. 5,218,704.

(51) **Int. Cl.**[7] **G06F 1/32**
(52) **U.S. Cl.** **713/322**; 713/601
(58) **Field of Search** 713/322, 323, 713/320, 300, 600, 601

(56) **References Cited**

U.S. PATENT DOCUMENTS

3,453,601 A	7/1969	Bogert et al.
3,623,017 A	11/1971	Lowell
3,868,647 A	2/1975	Zandvied
3,922,526 A	11/1975	Cochran 235/152
3,941,989 A	3/1976	McLaughlin et al.
4,137,563 A	1/1979	Tsunoda
4,217,637 A	8/1980	Faulkner et al.
4,254,475 A	3/1981	Cooney et al.
4,267,577 A	5/1981	Hashimoto et al.
4,279,020 A	7/1981	Christian et al.

(List continued on next page.)

FOREIGN PATENT DOCUMENTS

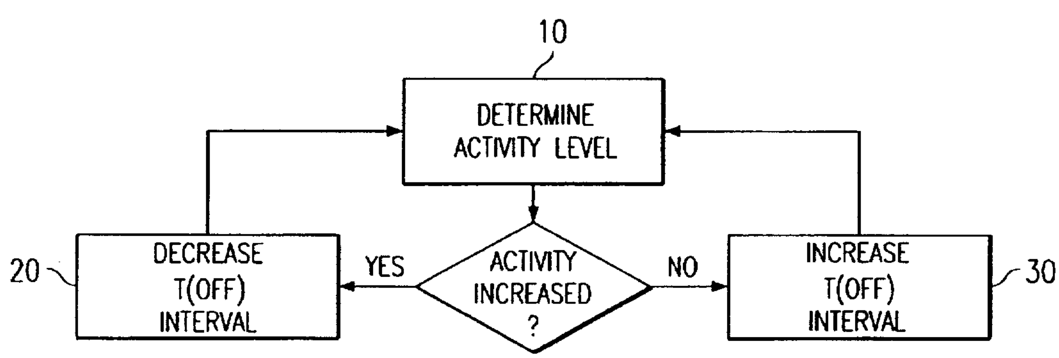
EP	0 349 726 B1	1/1990
EP	0 363 567 B1	4/1990
EP	8911349	4/1990
EP	0 349 726	10/1990

Primary Examiner—Glenn A. Auve
(74) *Attorney, Agent, or Firm*—Ronald O. Neerings; Wade James Brady, III; Frederick J. Telecky, Jr.

(57) **ABSTRACT**

A real-time power conservation apparatus and method for portable computers employs a monitor to determine whether a CPU may rest based upon a real-time sampling of the CPU activity level and to activate a hardware selector to carry out the monitor's determination. If the monitor determines the CPU may rest, the hardware selector reduces CPU clock time; if the CPU is to be active, the hardware selector returns the CPU to its previous high speed clock level. Switching back into full operation from its rest state occurs without a user having to request it and without any delay in the operation of the computer while waiting for the computer to return to a "ready" state. Furthermore, the monitor adjusts the performance level of the computer to manage power conservation in response to the real-time sampling of CPU activity. Such adjustments are accomplished within the CPU cycles and do not affect the user's perception of performance and do not affect any system application software executing on the computer.

38 Claims, 6 Drawing Sheets



U.S. PATENT DOCUMENTS

4,293,927	A	10/1981	Hoshii	
4,316,247	A	2/1982	Iwamoto	
4,317,180	A	2/1982	Lies	
4,317,181	A	2/1982	Teza et al.	
4,361,873	A	11/1982	Harper et al.	
4,381,552	A	4/1983	Nocilini et al.	
4,409,665	A	10/1983	Tubbs et al.	
4,590,553	A	5/1986	Noda	
4,612,418	A	9/1986	Takeda et al.	179/81 R
4,615,006	A	9/1986	Maejima et al.	
4,670,837	A	6/1987	Sheets	
4,686,386	A	8/1987	Tadao	307/269
4,698,748	A	10/1987	Juzswik et al.	
4,748,559	A	5/1988	Smith et al.	
4,758,945	A	7/1988	Remedi	
4,780,843	A	10/1988	Tietjen	
4,814,591	A	3/1989	Nara et al.	235/380
4,819,164	A	4/1989	Branson	
4,821,229	A	4/1989	Jauregui	
4,823,292	A	4/1989	Hillion	
4,823,309	A	4/1989	Kusaka et al.	
4,851,987	A	7/1989	Day	
4,870,570	A	9/1989	Satoh et al.	
4,893,271	A	1/1990	Davis et al.	
4,980,836	A	12/1990	Carter et al.	
5,025,387	A	6/1991	Frane	
5,083,266	A	1/1992	Watanabe	
5,086,387	A	2/1992	Arroyo et al.	
5,129,091	A	7/1992	Yorimoto et al.	
5,142,684	A	8/1992	Perry et al.	
5,167,024	A	11/1992	Smith	
5,179,693	A	1/1993	Kitamura et al.	
5,201,059	A	4/1993	Nguyen	
5,218,704	A	6/1993	Watts, Jr. et al.	
5,560,024	A	9/1996	Harper et al.	
6,173,409	B1 *	1/2001	Watts, Jr. et al.	713/322

* cited by examiner

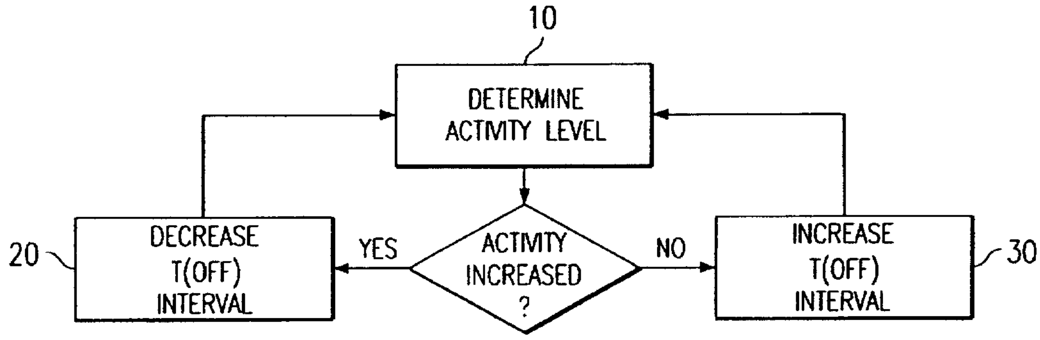


FIG. 1

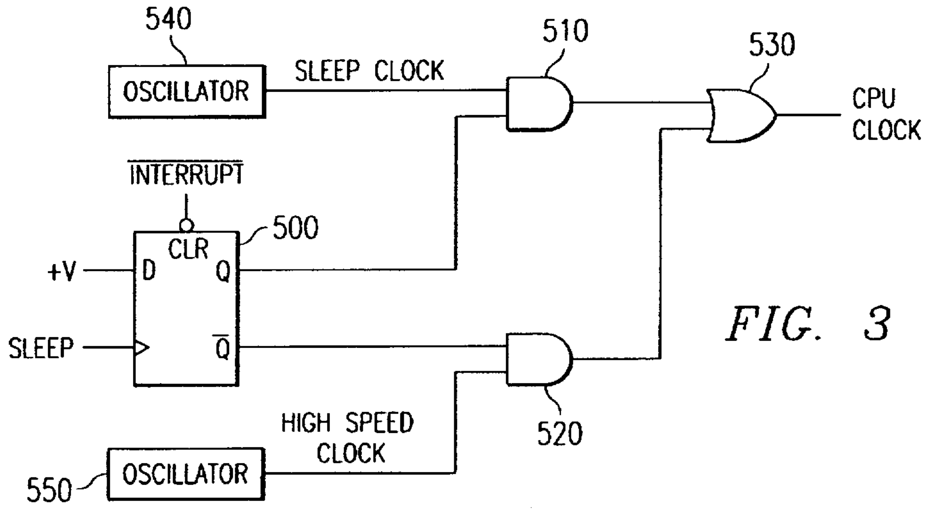


FIG. 3

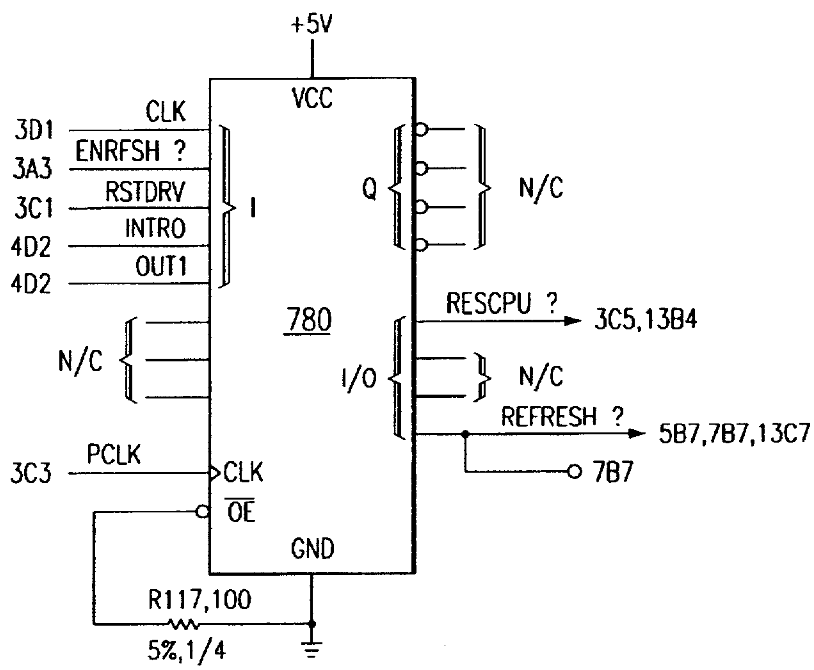
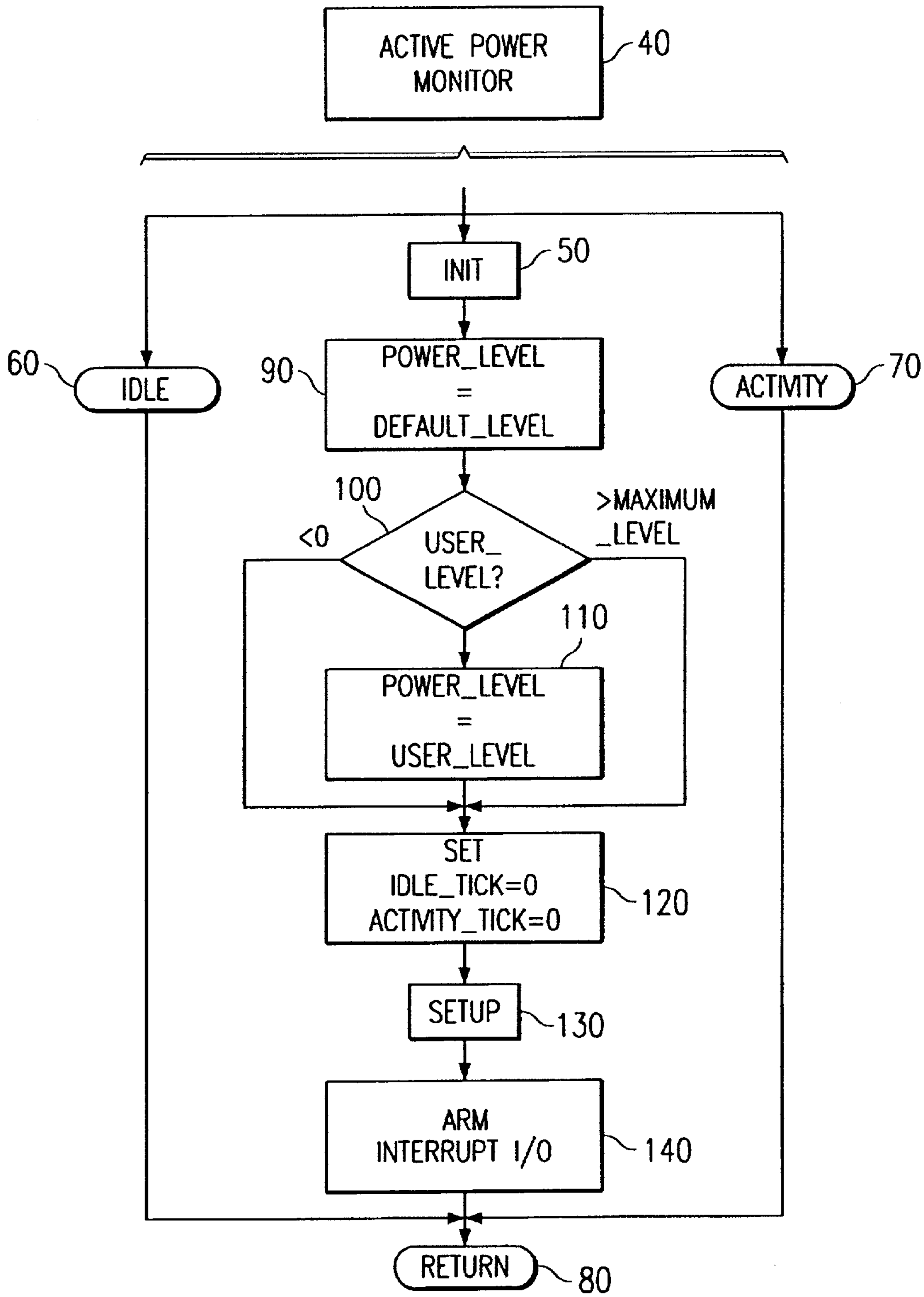


FIG. 5



ACTIVE POWER MONITOR
40
INIT
50
60
IDLE
90
POWER_LEVEL = DEFAULT_LEVEL
ACTIVITY
70
100
USER_ LEVEL?
<0
>MAXIMUM _LEVEL
110
POWER_LEVEL = USER_LEVEL
SET IDLE_TICK=0 ACTIVITY_TICK=0
120
SETUP
130
ARM INTERRUPT I/O
140
RETURN
80

FIG. 2a

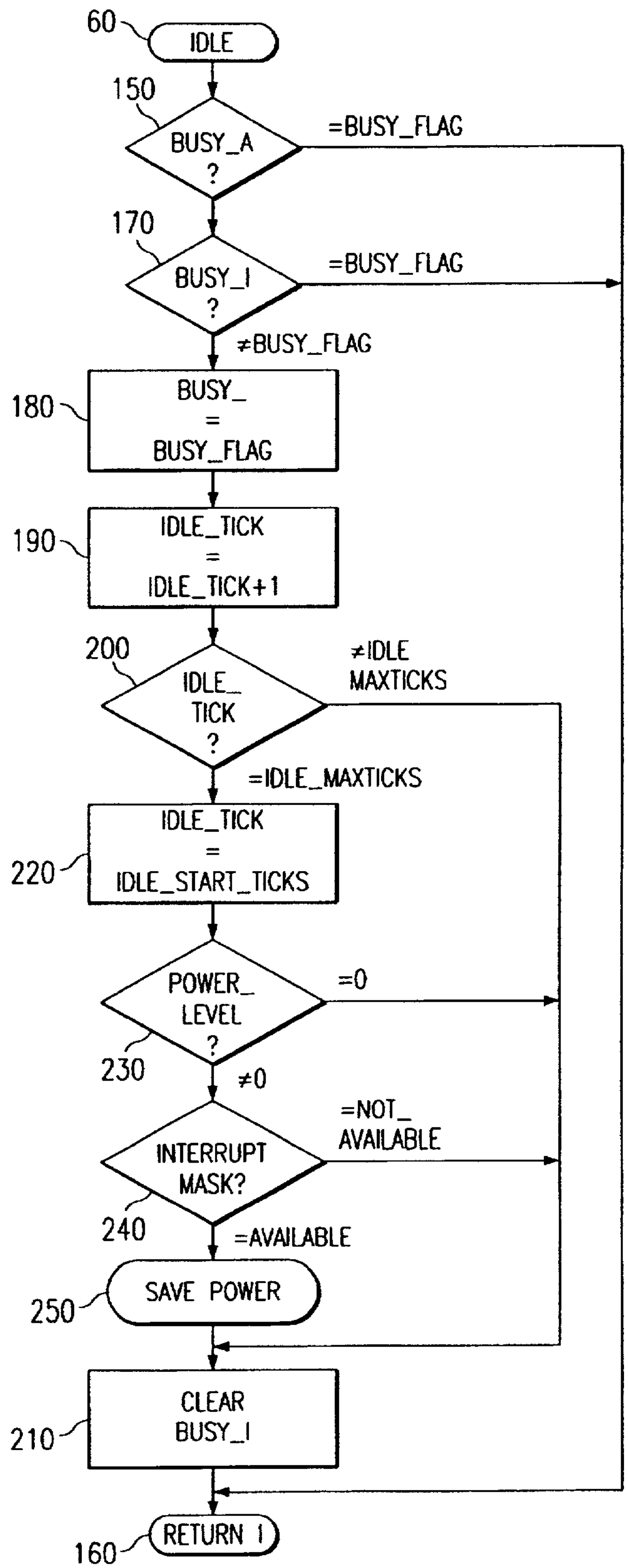

60
IDLE
150
BUSY_A
?
=BUSY_FLAG
170
BUSY_I
?
=BUSY_FLAG
≠BUSY_FLAG
180
BUSY_
=
BUSY_FLAG
190
IDLE_TICK
=
IDLE_TICK+1
200
IDLE_
TICK
?
≠IDLE
MAXTICKS
=IDLE_MAXTICKS
220
IDLE_TICK
=
IDLE_START_TICKS
POWER_
LEVEL
?
=0
230
≠0
INTERRUPT
MASK?
=NOT_
AVAILABLE
240
=AVAILABLE
250
SAVE POWER
210
CLEAR
BUSY_I
160
RETURN I

FIG. 2b

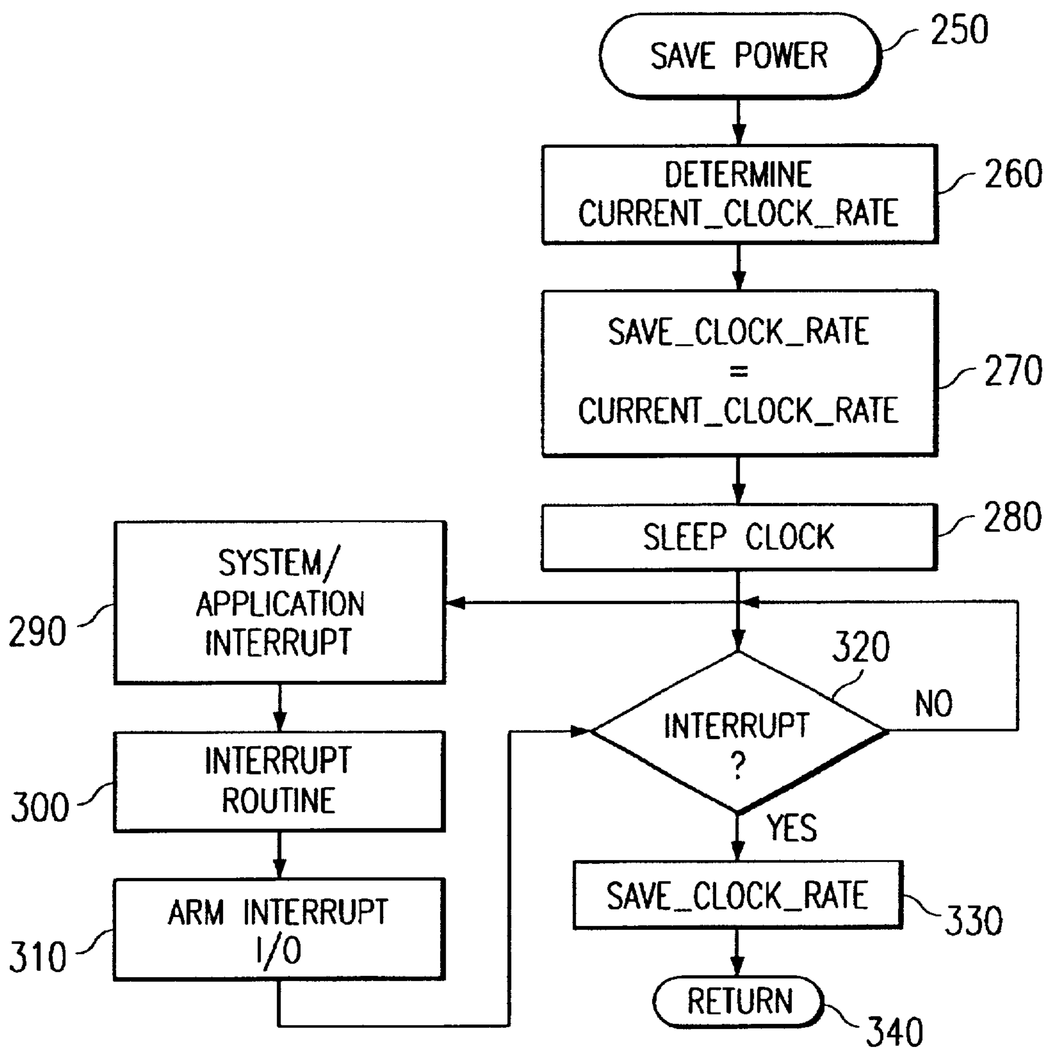

SAVE POWER
250
DETERMINE CURRENT_CLOCK_RATE
260
SAVE_CLOCK_RATE = CURRENT_CLOCK_RATE
270
SLEEP CLOCK
280
SYSTEM/ APPLICATION INTERRUPT
290
INTERRUPT ROUTINE
300
ARM INTERRUPT I/O
310
INTERRUPT ?
320
NO
YES
SAVE_CLOCK_RATE
330
RETURN
340

FIG. 2c

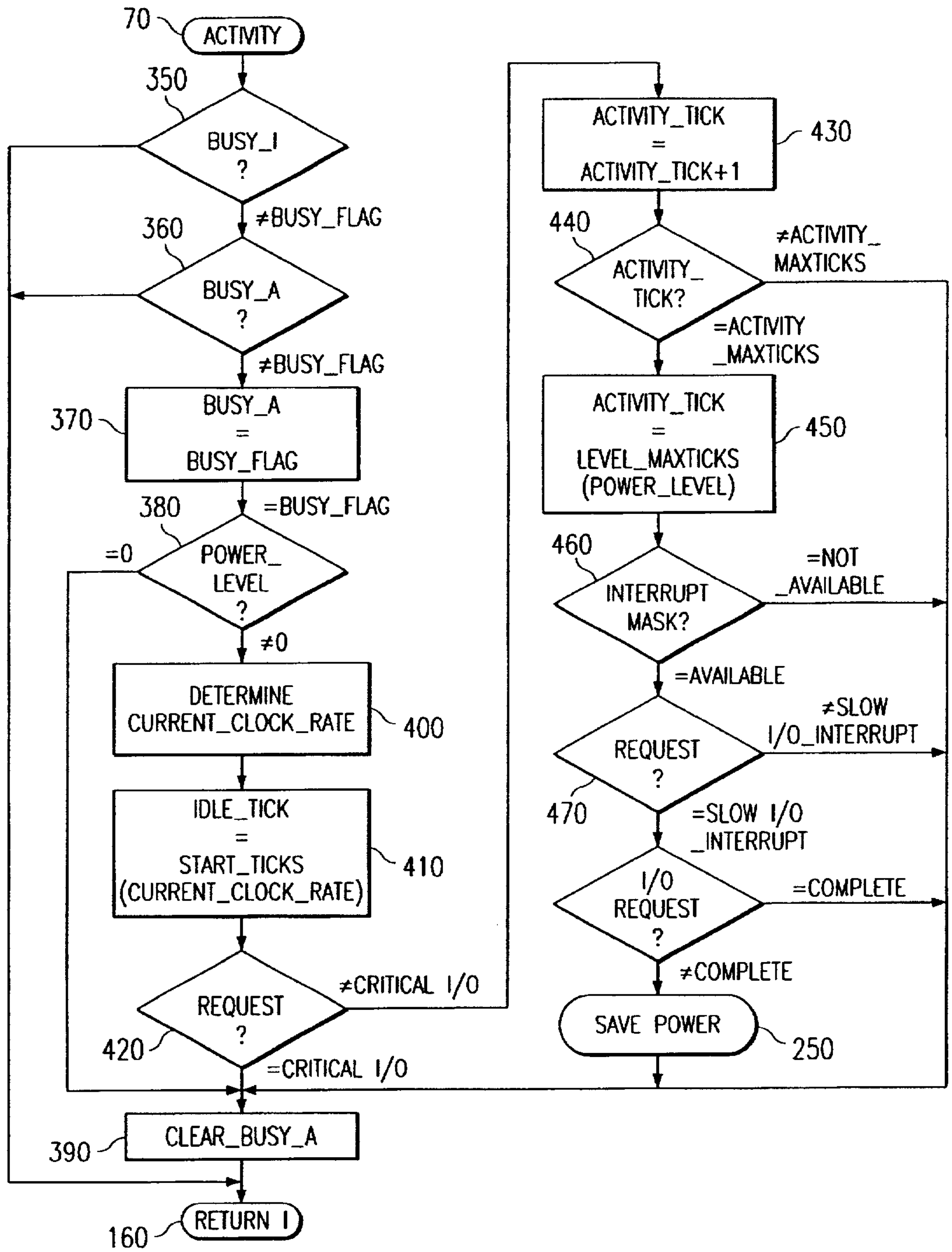
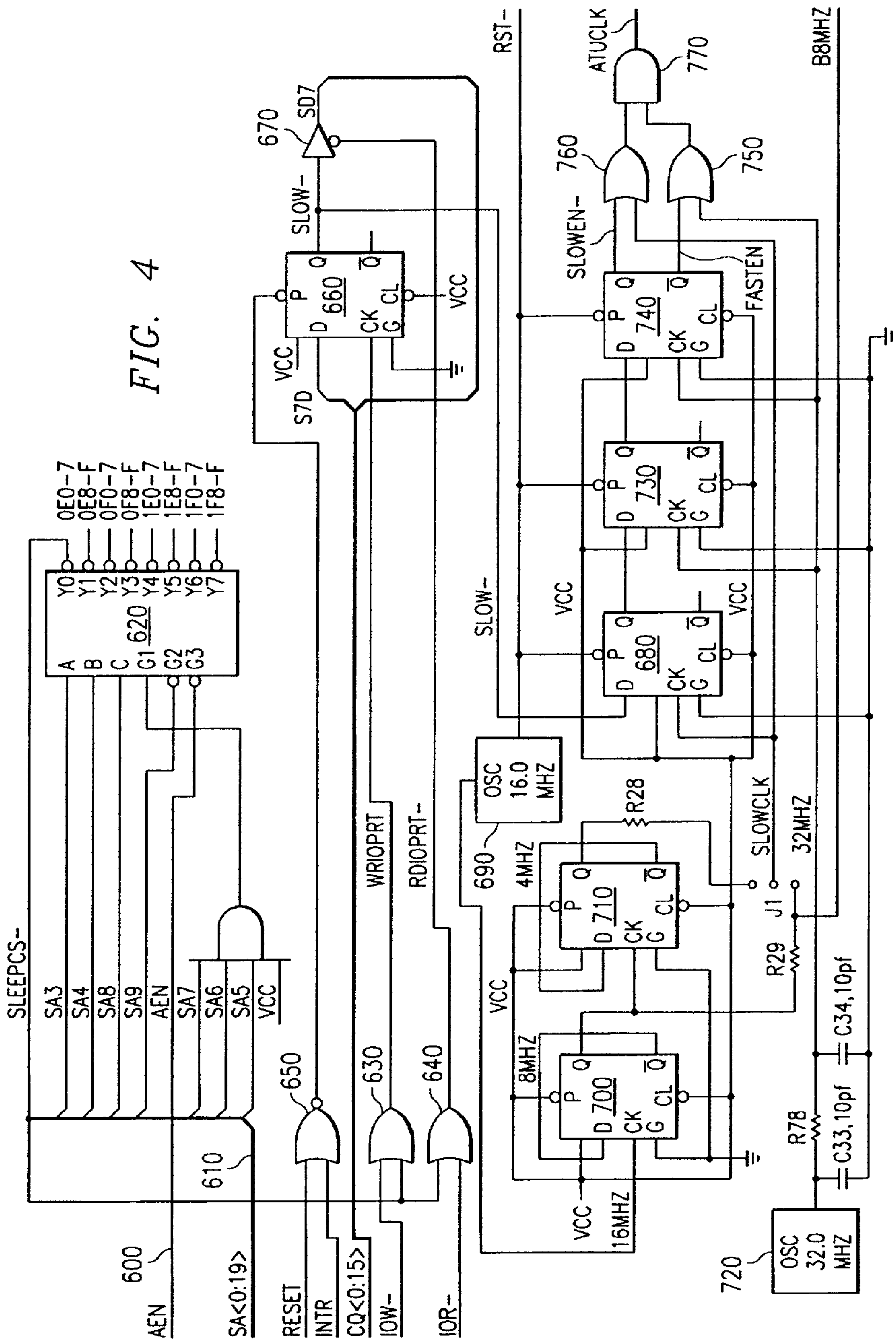

70
ACTIVITY
350
BUSY_I ?
≠BUSY_FLAG
360
BUSY_A ?
≠BUSY_FLAG
370
BUSY_A = BUSY_FLAG
380
=BUSY_FLAG
=0
POWER_ LEVEL ?
≠0
DETERMINE CURRENT_CLOCK_RATE
400
IDLE_TICK = START_TICKS (CURRENT_CLOCK_RATE)
410
REQUEST ?
≠CRITICAL I/O
420
=CRITICAL I/O
CLEAR_BUSY_A
390
RETURN I
160
ACTIVITY_TICK = ACTIVITY_TICK+1
430
440
ACTIVITY_ TICK?
≠ACTIVITY_ MAXTICKS
=ACTIVITY _MAXTICKS
ACTIVITY_TICK = LEVEL_MAXTICKS (POWER_LEVEL)
450
460
INTERRUPT MASK?
=NOT _AVAILABLE
=AVAILABLE
REQUEST ?
≠SLOW I/O_INTERRUPT
470
=SLOW I/O _INTERRUPT
I/O REQUEST ?
=COMPLETE
≠COMPLETE
SAVE POWER
250

FIG. 2d



AEN
SA<0:19>
RESET
INTR
CQ<0:15>
IOW-
IOR-
600
610
620
630
640
650
660
670
680
690
700
710
720
730
740
750
760
770
SLEEPCS-
SA3
SA4
SA8
SA9
SA7
SA6
SA5
VCC
0E0-7
0E8-F
0F0-7
0F8-F
1E0-7
1E8-F
1F0-7
1F8-F
WRIOPRT
RDIOPRT-
SLOW-
SD7
S7D
RST-
ATUCLK
B8MHZ
SLOWEN-
FASTEN
OSC 16.0 MHZ
OSC 32.0 MHZ
4MHZ
8MHZ
16MHZ
R28
R29
R78
J1
SLOWCLK
32MHZ
C34,10pf
C33,10pf

FIG. 4

REAL-TIME POWER CONSERVATION FOR ELECTRONIC DEVICE HAVING A PROCESSOR

This application is a Continuation of application Ser. No.09/392,205, filed Sep. 8, 1999,now U.S. Pat. No. 6,173, 409 which is a Continuation of application Ser. No. 08/023, 831, filed Apr. 12, 1993, now U.S. Pat. No. 6,006,336 which is a Continuation of application Ser. No. 07/429,270 filed Oct. 30, 1989, now U.S. Pat. No. 5,218,704.

BACKGROUND Of THE INVENTION

1. Field of the Invention

This invention relates to real-time computer power conservation, and more particularly to an apparatus and method for reduction of central processing unit (CPU) clock time based on the real-time activity level within the CPU of a portable computer.

2. Description of the Related Art

During the development stages of personal computers, the transportable or portable computer has become very popular. Such portable computer uses a large power supply and really represents a small desktop personal computer. Portable computers are smaller and lighter than a desktop personal computer and allow an user to employ the same software that can be used on a desktop computer.

The first generation "portable" computers only operated from an A/C wall power. As personal desktop computer development continued, battery-powered computers were designed. Furthermore, real portability became possible with the development of new display technology, better disk storage, and lighter components.

However, the software developed was designed to run on a desk top personal computers, with all the features of a computer, without regard to battery-powered portable computers that only had limited amounts of power available for short periods of time. No special considerations were made by the software, operating system (MS-DOS), Basic Input/ Output System (BIOS), or the third party application software to conserve power usage for these portable computers.

As more and more highly functional software packages were developed, desk top computer users experienced increased performance from the introductions of higher computational CPUs, increased memory, and faster high performance disk drives.

Unfortunately, portable computers continued to run only on A/C power or with large and heavy batteries. In trying to keep up with the performance requirements of the desk top computers, and the new software, expensive components were used to cut the power requirements. Even so, the heavy batteries still did not run very long. This meant users of portable computers had to settle for A/C operation or very short battery operation to have the performance that was expected from the third party software.

Portable computer designers stepped the performance down to 8088- and 8086-type processors to reduce the power consumption. The supporting circuits and CPU took less power to run and therefore, lighter batteries could be used. Unfortunately, the new software requiring 80286-type instructions, that did not exist in the older slower 8088/8086 CPUs, did not run.

In an attempt to design a portable computer that could conserve power, thereby yielding longer battery operation, smaller units, and less weight, some portable computer designers proceeded to reduce power consumption of a portable computer while user is not using the computer. For example, designers obtain a reduction in power usage by slowing or stopping the disk drive after some predetermined period of inactivity; if the disk drive is not being used, the disk drive is turned off, or simply placed into a standby mode. When the user ready to use the disk, the operator must wait until the disk drive is spins up and the computer system is ready again for full performance before the operator may proceed with the operation.

Other portable computer designers conserve power by turning the computer display off when the keyboard is not being used. However, in normal operation the computer is using full power. In other words, power conservation by this method is practical only when the user is not using the components of the system. It is very likely, however, that the user will turn the computer off when not in use.

Nevertheless, substantial power conservation while the operator is using the computer for meaningful work is needed. When the operator uses the computer, full operation of all components is required. During the intervals while the operator is not using the computer, however, the computer could be turned off or slowed down to conserve power consumption. It is critical to maintaining performance to determine when to slow the computer down or turn it off without disrupting the user's work, upsetting the third party software, or confusing the operating system, until operation is needed.

Furthermore, although an user can wait for the disk to spin up as described above, application software packages cannot wait for the CPU to "spin up" and get ready. The CPU must be ready when the application program needs to compute. Switching to full operation must be completed quickly and without the application program being affected. This immediate transition must be transparent to the user as well as to the application currently active. Delays cause user operational problems in response time and software compatability, as well as general failure by the computer to accurately execute a required program.

Other attempts at power conservation for portable computers include providing a "Shut Down" or "Standby Mode" of operation. The problem, again, is that the computer is not usable by the operator during this period. The operator could just as well turned off the power switch of the unit to save power. This type of power conservation only allows the portable computer to "shut down" and thereby save power if the operator forgets to turn off the power switch, or walks away from the computer for the programmed length of time. The advantage of this type of power conservation over just turning the power switch off/on is a much quicker return to full operation. However, this method of power conservation is still not real-time, intelligent power conservation while the computer is on and processing data which does not disturb the operating system, BIOS, and any third party application programs currently running on the computer.

Attempts to meet this need have been made by VLSI vendors in providing circuits that either turn off the clocks to the CPU when the user is not typing on the keyboard or wakes up the computer on demand when a keystroke occurred. Either of these approaches reduce power but the computer is dead (unusable) during this period. Background operations such as updating the system clock, communications, print spooling, and other like operations cannot be performed. Some existing portable computers employ these circuits. After a programmed period of no activity, the computer turns itself off. The operator must turn the machine on again but does not have to reboot the

operating system and application program. The advantage of this circuitry is, like the existing "shut down" operations, a quick return to full operation without restarting the computer. Nevertheless, this method only reduces power consumption when the user walks away from the machine and does not actually extend the operational life of the battery charge.

SUMMARY OF THE INVENTION

In view of the above problems associated with the related art, it is an object of the present invention to provide an apparatus and method for real-time conservation of power for computer systems without any real-time performance degradation, such conservation of power remaining transparent to the user.

Another object of the present invention is to provide an apparatus and method for predicting the activity level within a computer system and using the prediction for automatic power conservation.

Yet another object of the present invention is to provide an apparatus and method which allows user modification of automatic activity level predictions and using the modified predictions for automatic power conservation.

A further object of the present invention is to provide an apparatus and method for real-time reduction and restoration of clock speeds thereby returning the CPU to full processing rate from a period of inactivity which is transparent to software programs.

These objects are accomplished in a preferred embodiment of the present invention by an apparatus and method which determine whether a CPU may rest based upon the CPU activity level and activates a hardware selector based upon that determination. If the CPU may rest, or sleep, the hardware selector applies oscillations at a sleep clock level; if the CPU is to be active, the hardware selector applies oscillations at a high speed clock level.

The present invention examines the state of CPU activity, as well as the activity of both the operator and any application software program currently active. This sampling of activity is performed real-time, adjusting the performance level of the computer to manage power conservation and computer power. These adjustments are accomplished within the CPU cycles and do not affect the user's perception of performance.

Thus, when the operator for the third party software of the operating system/BIOS is not using the computer, the present invention will effect a quick turn off or slow down of the CPU until needed, thereby reducing the power consumption, and will promptly restore full CPU operation when needed without affecting perceived performance. This switching back into full operation from the "slow down" mode occurs without the user having to request it and without any delay in the operation of the computer while waiting for the computer to return to a "ready" state.

These and other features and advantages of the invention will be apparent to those skilled in the art from the following detailed description of a preferred embodiment, taken together with the accompanying drawings, in which:

DESCRIPTION OF THE DRAWINGS

FIG. 1 is a flowchart depicting the self-tuning aspect of a preferred embodiment of the present invention;

FIGS. 2*a*–2*d* are flowcharts depicting the active power conservation monitor employed by the present invention;

FIG. 3 is a simplified schematic diagram representing the active power conservation associated hardware employed by the present invention;

FIG. 4 is a schematic of the sleep hardware for one embodiment of the present invention; and

FIG. 5 is a schematic of the sleep hardware for another embodiment of the present invention.

DETAILED DESCRIPTION OF A PREFERRED EMBODIMENT

If the period of computer activity in any given system is examined, the CPU and associated components have a utilization percentage. If the user is inputing data from the keyboard, the time between keystrokes is very long in terms of CPU cycles. Many things can be accomplished by the computer during this time, such as printing a report. Even during the printing of a report, time is still available for additional operations such as background updating of a clock/calendar display. Even so, there is almost always spare time when the CPU is not being used. If the computer is turned off or slowed down during this spare time, then power consumption is obtained real-time. Such real-time power conservation extends battery operation life.

According to the preferred embodiment of the present invention, to conserve power under MS-DOS, as well as other operating systems such as OS/2, XENIX, and those for Apple computers, requires a combination of hardware and software. It should be noted that because the present invention will work in any system, while the implementation may vary slightly on a system-by-system basis, the scope of the present invention should therefore not be limited to computer systems operating under MS/DOS.

Slowing down or stopping the computer system components according to the preferred embodiment of the present invention, reduces power consumption, although the amount of power saved may vary. Therefore, according to the present invention, stopping the clock (where possible as some CPUs cannot have their clocks stopped) reduces the power consumption more than just slowing the clock.

In general, the number of operations (or instructions) per second may be considered to be roughly proportional to the processor clock:

$$\text{instructions/second}=\text{instructions/cycle}*\text{cycles/second}$$

Assuming for simplicity that the same instruction is repeatedly executed so that instructions/second is constant, the relationship can be expressed as follows:

$$Fq=K_1*Clk$$

where Fq is instructions/second, K_1 is constant equal to the instructions/cycle, and Clk equals cycles/second. Thus, roughly speaking, the rate of execution increases with the frequency of the CPU clock.

The amount of power being used at any given moment is also related to the frequency of the CPU clock and therefore to the rate of execution. In general this relationship can be expressed as follows:

$$P=K_2+(K_3*Clk)$$

where P is power in watts, K_2 is a constant in watts, K_3 is a constant and expresses the number of watt-seconds/cycle, and Clk equals the cycles/second of the CPU clock. Thus it can also be said that the amount of power being consumed at any given time increases as the CPU clock frequency increases.

Assume that a given time period T is divided into N intervals such that the power P was constant during each

interval. Then the amount of energy E expended during T would be given by:

$$E=P(1)\text{delta}T_1+P(2)\text{delta}T_2 \ldots +P(N)\text{delta}T_N$$

Further assume that the CPU clock "Clk" has only two states, either "ON" or "OFF". For the purposes of this discussion, the "ON" state represents the CPU clock at its maximum frequency, while the "OFF" state represents the minimum clock rate at which the CPU can operate (this may be zero for CPUs that can have their clocks stopped). For the condition in which the CPU clock is always "ON", each P(i) in the previous equation is equal and the total energy is:

$$\begin{aligned} E(\max) &= P(\text{on})*(\text{delta } T_1 + \text{delta } T_2 \ldots + \text{delta } T_N) \\ &= P(\text{on})*T \end{aligned}$$

This represents the maximum power consumption of the computer in which no power conservation measures are being used. If the CPU clock is "off" during a portion of the intervals, then there are two power levels possible for each interval. The P(on) represents the power being consumed when the clock in in its "ON" state, while P(off) represents the power being used when the clock is "OFF". If all of the time intervals in which the clock is "ON" is summed into the quantity "T(on)" and the "OFF" intervals are summed into "T(off)", then it follows:

$$T=T(\text{on})+T(\text{off})$$

Now the energy being used during period T can be written:

$$E=[P(\text{on})*T(\text{on})]+[P(\text{off})*T(\text{off})]$$

Under these conditions, the total energy consumed may be reduced by increasing the time intervals T(off). Thus, by controlling the periods of time the clock is in its "OFF" state, the amount of energy being used may be reduced. If the T(off) period is divided into a large number of intervals during the period T, then as the width of each interval goes to zero, energy consumption is at a maximum. Conversely, as the width of the T(off) intervals increase, the energy consumed decreases.

If the "OFF" intervals are arranged to coincide with periods during which the CPU is normally inactive, then the user cannot perceive any reduction in performance and overall energy consumption is reduced from the E(max) state. In order to align the T(off) intervals with periods of CPU inactivity, the CPU activity level is used to determine the width of the T(off) intervals in a closed loop. FIG. **1** depicts such a closed loop. The activity level of the CPU is determined at Step **10**. If this level is an increase over an immediately previous determination, the present invention decreases the T(off) interval (Step **20**) and returns to determine the activity level of the CPU again. If, on the other hand, this activity level is a decrease over an immediately previous determination, the present invention increases the T(off) interval (Step **30**) and proceeds to again determine the activity level of the CPU. Thus the T(off) intervals are constantly being adjusted to match the system activity level.

In any operating system, two key logic points exist: an IDLE, or "do nothing", loop within the operating system and an operating system request channel, usually available for services needed by the application software. By placing logic inline with these logic points, the type of activity request made by an application software can be evaluated, power conservation can be activated and slice periods determined. A slice period is the number of T(on) vs. T(off) intervals over time, computed by the activity level. An assumption may be made to determine CPU activity level: Software programs that need service usually need additional services and the period of time between service requests can be used to determine the activity level of any application software running on the computer and to provide slice counts for power conservation according to the present invention.

Once the CPU is interrupted during a power conservation slice (T(off)), the CPU will save the interrupted routine's state prior to vectoring to the interrupt software. Of course, since the power conservation software was operating during this slice, control will be returned to the active power conservation loop (monitor **40**) which simply monitors the CPU's clock to determine an exit condition for the power conservation mode, thereby exiting from T(off) to T(on) state. The interval of the next power conservation state is adjusted by the activity level monitored, as discussed above in connection with FIG. **1**. Some implementations can create an automatic exit from T(off) by the hardware logic, thereby forcing the power conservation loop to be exited automatically and executing an interval T(on).

More specifically, looking now at FIGS. *2a–2d*, which depict the active power conservation monitor **40** of the present invention. The CPU installs monitor **40** either via a program stored in the CPU ROM or loads it from an external device storing the program in RAM. Once the CPU has loaded monitor **40**, it continues to INIT **50** for system interrupt initialization, user configurational setup, and system/application specific initialization. IDLE branch **60** (more specifically set out in FIG. *2b*) is executed by a hardware or software interrupt for an IDLE or "do nothing" function. This type of interrupt is caused by the CPU entering either an IDLE or a "do nothing" loop (i.e., planned inactivity). The ACTIVITY branch **70** of the flowchart, more fully described below in relation to FIG. *2d*, is executed by a software or hardware interrupt due to an operating system or I/O service request, by an application program or internal operating system function. An I/O service request made by a program may, for example, be a disk I/O, read, print, load, etc. Regardless of the branch selected, control is eventually returned to the CPU operating system at RETURN **80**. The INIT branch **50** of this flowchart, shown in FIG. *2a*, is executed only once if it is loaded via program into ROM or is executed every time during power up if it is loaded from an external device and stored in the RAM. Once this branch of active power monitor **40** has been fully executed, whenever control is yielded from the operating system to the power conservation mode, either IDLE **60** or ACTIVITY **70** branches are selected depending on the type of CPU activity: IDLE branch **60** for power conservation during planned inactivity and ACTIVITY branch **70** for power conservation during CPU activity.

Looking more closely at INIT branch **50** , after all system interrupt and variables are initialized, the routine continues at Step **90** to set the Power_level equal to $\text{DEFAULT}_{13}\text{LEVEL}$. In operating systems where the user has input control for the Power_level, the program at Step **100** checks to see if a User_level has been selected. If the User_level is less than zero or greater than the MAXIMUM_LEVEL, the system uses the DEFAULT_LEVEL. Otherwise, it continues onto Step **110** where it modifies the Power_level to equal the User_level.

According to the preferred embodiment of the present invention, the system at Step **120** sets the variable Idle_tick to zero and the variable Activity_tick to zero. Under an

MS/DOS implementation, Idle_tick refers to the number of interrupts found in a "do nothing" loop. Activity_tick refers to the number of interrupts caused by an activity interrupt which in turn determines the CPU activity level. Tick count represents a delta time for the next interrupt. Idle_tick is a constant delta time from one tick to another (interrupt) unless overwritten by a software interrupt. A software interrupt may reprogram delta time between interrupts.

After setting the variables to zero, the routine continues on to Setup **130** at which time any application specific configuration fine-tuning is handled in terms of system-specific details and the system is initialized. Next the routine arms the interrupt I/O (Step **140**) with instructions to the hardware indicating the hardware can take control at the next interrupt. INIT branch **50** then exits to the operating system, or whatever called the active power monitor originally, at RETURN **80**.

Consider now IDLE branch **60** of active power monitor **40**, more fully described at FIG. 2*b*. In response to a planned inactivity of the CPU, monitor **40** (not specifically shown in this Figure) checks to see if entry into IDLE branch **60** is permitted by first determining whether the activity interrupt is currently busy. If Busy_A equals BUSY_FLAG (Step **150**), which is a reentry flag, the CPU is busy and cannot now be put to sleep. Therefore, monitor **40** immediately proceeds to RETURN I **160** and exits the routine. RETURN I **160** is an indirect vector to the previous operating system IDLE vector interrupt for normal processing stored before entering monitor **40**. (I.e., this causes an interrupt return to the last chained vector.)

If the Busy_A interrupt flag is not busy, then monitor **40** checks to see if the Busy Idle interrupt flag, Busy_I, equals BUSY_FLAG (Step **170**). If so, this indicates the system is already in IDLE branch **60** of monitor **40** and therefore the system should not interrupt itself. If Busy_I=BUSY_FLAG, the system exits the routine at RETURN_I indirect vector **160**.

If, however, neither the Busy_A reentry flag or the Busy_I reentry flag have been set, the routine sets the Busy_I flag at Step **180** for reentry protection (Busy_I=BUSY_FLAG). At Step **190** Idle_tick is incremented by one. Idle_tick is the number of T(on) before a T(off) interval and is determined from IDLE interrupts, setup interrupts and from CPU activity level. Idle_tick increments by one to allow for smoothing of events, thereby letting a critical I/O activity control smoothing.

At Step **200** monitor **40** checks to see if Idle_tick equals IDLE_MAXTICKS. IDLE_MAXTICKS is one of the constants initialized in Setup **130** of INIT branch **50**, remains constant for a system, and is responsible for self-tuning of the activity level. If Idle_tick does not equal IDLE MAXTICKS, the Busy_I flag is cleared at Step **210** and exits the loop proceeding to the RETURN I indirect vector **160**. If, however, Idle_tick equals IDLE_MAXTICKS, Idle_tick is set equal to IDLE START_TICKS (Step **220**). IDLE_START TICKS is a constant which may or may not be zero (depending on whether the particular CPU can have its clock stopped). This step determines the self-tuning of how often the rest of the sleep functions may be performed. By setting IDLE START_TICKS equal to IDLE_MAXTICKS minus one, a continuous T(off) interval is achieved. At Step **230**, the Power_level is checked. If it is equal to zero, the monitor clears the Busy_I flag (Step **210**), exits the routine at RETURN I **160**, and returns control to the operating system so it may continue what it was originally doing before it entered active power monitor **40**.

If, however, the Power_level does not equal zero at Step **240**, the routine determines whether an interrupt mask is in place. An interrupt mask is set by the system/application software, and determines whether interrupts are available to monitor **40**. If interrupts are NOT_AVAILABLE, the Busy_I reentry flag is cleared and control is returned to the operating system to continue what it was doing before it entered monitor **40**. Operating systems, as well as application software, can set T(on) interval to yield a continuous T(on) state by setting the interrupt mask equal to NOT_AVAILABLE.

Assuming an interrupt is AVAILABLE, monitor **40** proceeds to the SAVE POWER subroutine **250** which is fully executed during one T(off) period established by the hardware state. (For example, in the preferred embodiment of the present invention, the longest possible interval could be **18** ms, which is the longest time between two ticks or interrupts from the real-time clock.) During the SAVE POWER subroutine **250**, the CPU clock is stepped down to a sleep clock level.

Once a critical I/O operation forces the T(on) intervals, the IDLE branch **60** interrupt tends to remain ready for additional critical I/O requests. As the CPU becomes busy with critical I/O, less T(off) intervals are available. Conversely, as critical I/O requests decrease, and the time intervals between them increase, more T(off) intervals are available. IDLE branch **60** is a self-tuning system based on feedback from activity interrupts and tends to provide more T(off) intervals as the activity level slows. As soon as monitor **40** has completed SAVE POWER subroutine **250**, shown in FIG. 2*c* and more fully described below, the Busy_I reentry flag is cleared (Step **210**) and control is returned at RETURN I **160** to whatever operating system originally requested monitor **40**.

Consider now FIG. 2*c*, which is a flowchart depicting the SAVE POWER subroutine **250**. Monitor **40** determines what the I/O hardware high speed clock is at Step **260**. It sets the CURRENT_CLOCK_RATE equal to the relevant high speed clock and saves this value to be used for CPUs with multiple level high speed clocks. Thus, if a particular CPU has 12 MHz and 6 MHz high speed clocks, monitor **40** must determine which high speed clock the CPU is at before monitor **40** reduces power so it may reestablish the CPU at the proper high speed clock when the CPU awakens. At Step **270**, the Save_clock_rate is set equal to the CURRENT_CLOCK_RATE determined. Save_clock_rate **270** is not used when there is only one high speed clock for the CPU. Monitor **40** now continues to SLEEPCLOCK **280**, where a pulse is sent to the hardware selector (shown in FIG. 3) to put the CPU clock to sleep (i.e., lower or stop its clock frequency). The I/O port hardware sleep clock is at much lower oscillations than the CPU clock normally employed.

At this point either of two events can happen. A system/application interrupt may occur or a real-time clock interrupt may occur. If a system/application interrupt **290** occurs, monitor **40** proceeds to interrupt routine **300**, processing the interrupt as soon as possible, arming interrupt I/O at Step **310**, and returning to determine whether there has been an interrupt (Step **320**). Since in this case there has been an interrupt, the Save_clock_rate is used (Step **330**) to determine which high speed clock to return the CPU to and SAVE POWER subroutine **250** is exited at RETURN **340**. If, however, a system/application interrupt is not received, the SAVE POWER subroutine **250** will continue to wait until a real-time clock interrupt has occurred (Step **320**). Once such an interrupt has occurred, SAVE POWER subroutine **250** reestablishes the CPU at the stored Save-clock-rate. If the sleep clock rate was not stopped, in other words, the sleep clock rate was not zero, control is passed at a slow clock and

SAVE POWER subroutine **250** will execute interrupt loop **320** several times. If however, control is passed when the sleep clock rate was zero, in other words, there was no clock, the SAVE POWER subroutine **250** will execute interrupt loop **320** once before returning the CPU clock to the Save_clock_rate **330** and exiting (Step (**340**).

Consider now FIG. 2*d* which is a flowchart showing ACTIVITY branch **70** triggered by an application/system activity request via an operating system service request interrupt. ACTIVITY branch **70** begins with reentry protection. Monitor **40** determines at Step **350** whether Busy_I has been set to BUSY_FLAG. If it has, this means the system is already in IDLE branch **60** and cannot be interrupted. If Busy_I=BUSY_FLAG, monitor **40** exits to RETURN I **160**, which is an indirect vector to an old activity vector interrupt for normal processing, via an interrupt vector after the operating system performs the requested service.

If however, the Busy_I flag does not equal BUSY_FLAG, which means IDLE branch **60** is not being accessed, monitor **40** determines at Step **360** if the BUSY_A flag has been set equal to BUSY_FLAG. If so, control will be returned to the system at this point because ACTIVITY branch **70** is already being used and cannot be interrupted. If the Busy_A flag has not been set, in other words, Busy A does not equal BUSY FLAG, monitor **40** sets Busy_A equal to BUSY FLAG at Step **370** so as not to be interrupted during execution of ACTIVITY branch **70**. At Step **380** the Power_level is determined. If Power_level equals zero, monitor **40** exits ACTIVITY branch **70** after clearing the Busy_A reentry flag (Step **390**). If however, the Power_level does not equal zero, the CURRENT_CLOCK RATE of the I/O hardware is next determined. As was true with Step **270** of FIG. 2C, Step **400** of FIG. 2*d* uses the CURRENT_CLOCK_RATE if there are multiple level high speed clocks for a given CPU. Otherwise, CURRENT_CLOCK_RATE always equals the CPU high speed clock. After the CURRENT_CLOCK_RATE is determined (Step **400**), at Step **410** Idle_tick is set equal to the constant START_TICKS established for the previously determined CURRENT_CLOCK RATE. T(off) intervals are established based on the current high speed clock that is active.

Monitor **40** next determines that a request has been made. A request is an input by the application software running on the computer, for a particular type of service needed. At Step **420**, monitor **40** determines whether the request is a CRITICAL I/O. If the request is a CRITICAL I/O, it will continuously force T(on) to lengthen until the T(on) is greater than the T(off), and monitor **40** will exit ACTIVITY branch **70** after clearing the Busy_A reentry flag (Step **390**). If, on the other hand, the request is not a CRITICAL I/O, then the Activity_tick is incremented by one at Step **430**. It is then determined at Step **440** whether the Activity tick now equals ACTIVITY_MAXTICKS. Step **440** allows a smoothing from a CRITICAL I/O, and makes the system ready from another CRITICAL I/O during Activity_tick T(on) intervals. Assuming Activity tick does not equal ACTIVITY_MAXTICKS, ACTIVITY branch **70** is exited after clearing the Busy A reentry flag (Step **390**). If, on the other hand, the Activity_tick equals constant ACTIVITY_MAXTICKS, at Step **450** Activity_tick is set to the constant LEVEL_MAXTICKS established for the particular Power_level determined at Step **380**.

Now monitor **40** determines whether an interrupt mask exists (Stop **460**). An interrupt mask is set by system/application software. Setting it to NOT_AVAILABLE creates a continuous T(on) state. If the interrupt mask equals NOT_AVAILABLE, there are no interrupts available at this time and monitor **40** exits ACTIVITY branch **70** after clearing the Busy_A reentry flag (Step **390**). If, however, an interrupt is AVAILABLE, monitor **40** determines at Step **470** whether the request identified at Step **420** was for a SLOW I/O_INTERRUPT. SLOW I/O requests may have a delay until the I/O device becomes "ready". During the "make ready" operation, a continuous T(off) interval may be set up and executed to conserve power. Thus, if the request is not a SLOW I/O_INTERRUPT, ACTIVITY branch **70** is exited after clearing the Busy_A reentry flag (Step **390**). If, however, the request is a SLOW I/O_INTERRUPT, and time yet exists before the I/O device becomes "ready", monitor **40** then determines at Step **480** whether the I/O request is COMPLETE (i.e., is I/O device ready?). If the I/O device is not ready, monitor **40** forces T(off) to lengthen, thereby forcing the CPU to wait, or sleep, until the SLOW I/O device is ready. At this point it has time to save power and ACTIVITY branch **70** enters SAVE POWER subroutine **250** previously described in connection with to FIG. 2C. If, however, the I/O request is COMPLETE, control is returned to the operating system subsequently to monitor **40** exiting ACTIVITY branch **70** after clearing Busy_A reentry flag (Step **390**).

Self-tuning is inherent within the control system of continuous feedback loops. The software of the present invention can detect when CPU activity is low and therefore when the power conservation aspect of the present invention may be activated. Once the power conservation monitor is activated, a prompt return to full speed CPU clock operation within the interval is achieved so as to not degrade the performance of the computer. To achieve this prompt return to full speed CPU clock operation, the preferred embodiment of the present invention employs some associated hardware.

Looking now at FIG. **3** which shows a simplified schematic diagram representing the associated hardware employed by the present invention for active power conservation. When monitor **40** (not shown) determines the CPU is ready to sleep, it writes to I/O port (not shown) which causes a pulse on the SLEEP line. The rising edge of this pulse on the SLEEP line causes flip flop **500** to clock a high to Q and a low to Q-. This causes the AND/OR logic (AND gates **510**, **520**; OR gate **530**) to select the pulses travelling the SLEEP CLOCK line from SLEEP CLOCK oscillator **540** to be sent to and used by the CPU CLOCK. SLEEP CLOCK oscillator **540** is a slower clock than the CPU clock used during normal CPU activity. The high coming from the Q of flip flop **500** ANDed (**510**) with the pulses coming from SLEEP CLOCK oscillator **540** is ORed (**530**) with the result of the low on the Q- of flip flop **500** ANDed (**520**) with the pulse generated along the HIGH SPEED CLOCK line by the HIGH SPEED CLOCK oscillator **550** to yield the CPU CLOCK. When the I/O port designates SLEEP CLOCK, the CPU CLOCK is then equal to the SLEEP CLOCK oscillator **540** value. If, on the other hand, an interrupt occurs, an interrupt- value clears flip flop **500**, thereby forcing the AND/OR selector (comprising **510**, **520** and **530**) to choose the HIGH SPEED CLOCK value, and returns the CPU CLOCK value to the value coming from HIGH SPEED CLOCK oscillator **550**. Therefore, during any power conservation operation on the CPU, the detection of any interrupt within the system will restore the CPU operation at full clock rate prior to vectoring and processing the interrupt.

It should be noted that the associated hardware needed, external to each of the CPUs for any given system, may be different depending upon the operating system used,

whether the CPU can be stopped, etc. Nevertheless, the scope of the present invention should not be limited by possible system specific modifications needed to permit the present invention to actively conserve power in the numerous available portable computer systems. For example two actual implementations are shown in FIGS. **4** and **5**, discussed below.

Many VSLI designs today allow for clock switching of the CPU speed. The logic to switch from a null clock or slow clock to a fast clock logic is the same as that which allows the user to change speeds by a keyboard command. The added logic of monitor **40** working with such switching logic, causes an immediate return to a fast clock upon detection of any interrupt. This simple logic is the key to the necessary hardware support to interrupt the CPU and thereby allow the processing of the interrupt at full speed.

The method to reduce power consumption under MS-DOS employs the MS-DOS IDLE loop trap to gain access to the "do nothing" loop. The IDLE loop provides special access to application software and operating system operations that are in a state of IDLE or low activity. Careful examination is required to determine the activity level at any given point within the system. Feedback loops are used from the interrupt 21H service request to determine the activity level. The prediction of activity level is determined by interrupt 21H requests, from which the present invention thereby sets the slice periods for "sleeping" (slowing down or stopping) the CPU. An additional feature allows the user to modify the slice depending on the activity level of interrupt 21H.

Looking now at FIG. **4**, which depicts a schematic of an actual sleep hardware implementation for a system such as the Intel 80386 (CPU cannot have its clock stopped). Address enable bus **600** and address bus **610** provide CPU input to demultiplexer **620**. The output of demultiplexer **620** is sent along SLEEPCS- and provided as input to OR gates **630**,**640**. The other inputs to OR gates **630**,**640** are the I/O write control line and the I/O read control line, respectively. The outputs of these gates, in addition to NOR gate **650**, are applied to D flip flop **660** to decode the port. "INTR" is the interrupt input from the I/O port (peripherals) into NOR gate **650**, which causes the logic hardware to switch back to the high speed clock. The output of flip flop **660** is then fed, along with the output from OR gate **630**, to tristate buffer **670** to enable it to read back what is on the port. All of the above-identified hardware is used by the read/write I/O port (peripherals) to select the power saving "Sleep" operation. The output "SLOW-" is equivalent to "SLEEP" in FIG. **2**, and is inputted to flip flop **680**, discussed later.

The output of SLEEP CLOCK oscillator **690** is divided into two slower clocks by D flip flops **700**,**710**. In the particular implementation shown in FIG. **4**, 16 MHz sleep clock oscillator **690** is divided into 4 MHz and 8 MHZ clocks. Jumper **J1** selects which clock is to be the "SLEEP CLOCK".

In this particular implementation, high speed clock oscillator **720** is a 32 MHz oscillator, although this particular speed is not a requirement of the present invention. The 32 MHz oscillator is put in series with a resistor (for the implementation shown, 33 ohms), which is in series with two parallel capacitors (10 pF). The result of such oscillations is tied to the clocks of D flip flops **730**,**740**.

D flip flops **680**,**730**,**740** are synchronizing flip flops;

680,**730** were not shown in the simplified sleep hardware of FIG. **2**. These flip flops are used to ensure the clock switch occurs only on clock edge. As can be seen in FIG. **4**, as with flip flop **500** of FIG. **2**, the output of flip flop **740** either activates OR gate **750** or OR gate **760**, depending upon whether the CPU is to sleep ("FASTEN-") or awaken ("SLOWEN-").

OR gates **750**,**760** and AND gate **770** are the functional equivalents to the AND/OR selector of FIG. **2**. They are responsible for selecting either the "slowclk" (slow clock, also known as SLEEP CLOCK) or high speed clock (designated as 32 MHz on the incoming line). In this implementation, the Slow clock is either 4 MHz or 8 MHz, depending upon jumper J1, and the high speed clock is 32 MHz. The output of AND gate **770** (ATUCLK) establishes the rate of the CPU clock, and is the equivalent of CPU CLOCK of FIG. **2**.

Consider now FIG. **5**, which depicts a schematic of another actual sleep hardware implementation for a system such as the Intel 80286 (CPU can have its clock stopped) The Western Digital FE3600 VLSI is used for the speed switching with a special external PAL **780** to control the interrupt gating which wakes up the CPU on any interrupt. The software power conservation according to the present invention monitors the interrupt acceptance, activating the next P(i)deltaT$_i$ interval after the interrupt.

Any interrupt request to the CPU will return the system to normal operation. An interrupt request ("INTRQ") to the CPU will cause the PAL to issue a Wake Up signal on the RESCPU line to the FE3001 (not shown) which in turn enables the CPU and the DMA clocks to bring the system back to its normal state. This is the equivalent of the "INTERRUPT-" of FIG. **2**. Interrupt Request is synchronized to avoid confusing the state machine so that Interrupt (INTDET) will only be detected while the cycle is active. The rising edge of RESCPU will wake up the FE 3001 which in turn releases the whole system from the Sleep Mode.

Implementation for the 386SX is different only in the external hardware and software power conservation loop. The software loop will set external hardware to switch to the high speed clock on interrupt prior to vectoring the interrupt. Once return is made to the power conservation software, the high speed clock cycle will be detected and the hardware will be reset for full clock operation.

Implementation for OS/2 uses the "do nothing" loop programmed as a THREAD running in background operation with low priority. Once the THREAD is activated, the CPU sleep, or low speed clock, operation will be activated until an interrupt occurs thereby placing the CPU back to the original clock rate.

Although interrupts have been employed to wake up the CPU in the preferred embodiment of the present invention, it should be realized that any periodic activity within the system, or applied to the system, could also be used for the same function.

While several implementations of the preferred embodiment of the invention has been shown and described, various modifications and alternate embodiments will occur to those skilled in the art. Accordingly, it is intended that the invention be limited only in terms of the appended claims.

We claim:

1. An apparatus, comprising:

a processor having a monitor for measuring the relative amount of idle time within said processor, results of said measuring being used by said processor for providing a signal for circuitry for selectively modifying a clock signal being sent to said processor.

2. The apparatus of claim **1**, wherein said monitor is a program installed on said processor.

3. The apparatus of claim **1**, wherein said circuitry is external to said processor.

4. An apparatus, comprising:

a processor having a monitor for measuring the relative amount of idle time within said processor, results of said measuring being used by said processor for providing a signal for circuitry for selectively modifying a clock signal being sent to said processor in response to usage of said processor being below a preselected level.

5. An apparatus, comprising:

a processor having a monitor for measuring the relative amount of idle time within said processor, results of said measuring being used by said processor for providing a signal for circuitry for selectively modifying a clock signal being sent to said processor to reduce the idle time in said CPU.

6. The apparatus of claim 5, wherein said monitor inhibits the modification of said clock signal while said processor is processing critical I/O.

7. The apparatus of claim 5, wherein said processor sends signals to the circuitry requesting the circuitry to demodify the clock signal being sent to the processor in response to said monitor detecting a critical I/O request.

8. The apparatus of claim 1, wherein said monitor is self-tuning.

9. The apparatus of claim 8, wherein said monitor uses a control system of continuous feedback loops.

10. An apparatus, comprising:

a processor having a monitor for measuring the relative amount of idle time within said processor, results of said measuring being used by said processor for providing a signal for circuitry for selectively modifying a clock signal being sent to said processor to minimize the relative amount of idle time in said processor.

11. An apparatus, comprising:

a processor having a monitor for measuring the relative amount of activity time within said processor, results of said measuring being used by said processor for providing a signal for circuitry for selectively modifying a clock signal being sent to said processor.

12. The apparatus of claim 11, wherein said monitor inhibits the modification of said clock signal while said processor is processing critical I/O.

13. The apparatus of claim 11, wherein said processor sends signals to the circuitry requesting the circuitry to demodify the clock signal being sent to the processor in response to said monitor detecting a critical I/O request.

14. The apparatus of claim 11, wherein said monitor is self-tuning.

15. The apparatus of claim 14, wherein said monitor uses a control system of continuous feedback loops.

16. The apparatus of claim 11, wherein said monitor is a program installed on said processor.

17. The apparatus of claim 11, wherein said circuitry is external to said processor.

18. An apparatus, comprising:

a processor having a monitor for measuring the relative amount of activity time within said processor, results of said measuring being used by said processor for providing a signal for circuitry for selectively modifying a clock signal being sent to said processor in response to usage of said processor being below a preselected level.

19. An apparatus, comprising:

a processor having a monitor for measuring the relative amount of activity time within said processor, results of said measuring being used by said processor for providing a signal for circuitry for selectively modifying a clock signal being sent to said processor to control the amount of activity time in said processor.

20. An apparatus, comprising:

a processor having a monitor for measuring the relative amount of activity time within said processor, results of said measuring being used by said processor for providing a signal for circuitry for selectively modifying a clock signal being sent to said processor to optimize the activity time within said CPU in response to usage of said processor being below a preselected level.

21. An apparatus, comprising:

a processor having a monitor for measuring the relative amount of idle time and activity time within said processor, results of said measuring being used by said processor for providing a signal for circuitry for selectively modifying a clock signal being sent to said processor.

22. The apparatus of claim 21, wherein said monitor is a program installed on said processor.

23. The apparatus of claim 21, wherein said circuitry is external to said processor.

24. An apparatus, comprising:

a processor having a monitor for measuring the relative amount of idle time and activity time within said processor, results of said measuring being used by said processor for providing a signal for circuitry for selectively modifying a clock signal being sent to said processor in response to usage of said processor being below a preselected level.

25. An apparatus, comprising:

a processor having a monitor for measuring the relative amount of idle time and activity time within said processor, results of said measuring being used by said processor for providing a signal for circuitry for selectively modifying a clock signal being sent to said processor to control the amount of idle time and activity time in said CPU.

26. The apparatus of claim 25, wherein said monitor inhibits the modification of said clock signal while said processor is processing critical I/O.

27. The apparatus of claim 25, wherein said processor sends signals to the circuitry requesting the circuitry to demodify the clock signal being sent to the processor in response to said monitor detecting a critical I/O request.

28. The apparatus of claim 25, wherein said monitor is self-tuning.

29. The apparatus of claim 28, wherein said monitor uses a control system of continuous feedback loops.

30. An apparatus, comprising:

a processor having a monitor for measuring the relative amount of idle time and activity time within said processor, results of said measuring being used by said processor for providing a signal for circuitry for selectively modifying a clock signal being sent to said processor to control the amount of idle time and activity time in said processor in response to a utilization percentage of said processor being below a preselected level.

31. An apparatus, comprising:

a processor having a monitor for measuring the utilization of said processor, results of said measuring being used by said processor for providing a signal for circuitry for selectively modifying a clock signal being sent to said processor to control a utilization percentage of said processor.

32. An apparatus, comprising:

a processor coupled to a clock and having a monitor for measuring the relative amount of idle time and activity

time within said processor, results of said measuring being used by said processor for providing a signal for circuitry for controlling periods of time said clock is in an OFF state, the length of said periods of time said clock is in an OFF state being appropriate to allow said processor to operate at an efficient utilization percentage.

33. The apparatus of claim **32**, wherein energy consumption in said processor is at a maximum when the length of each period of time said clock is in an OFF state is at zero.

34. The apparatus of claim **32**, wherein energy consumption in said processor decreases as the length of each period of time said clock is in an OFF state increases.

35. The apparatus of claim **32**, wherein said periods of time said clock is in an OFF state are constantly being adjusted to optimize said utilization percentage of said processor.

36. The apparatus of claim **32**, wherein said OFF state represents the minimum clock rate at which said processor can operate.

37. The apparatus of claim **32**, wherein said minimum clock rate may be zero for processors that can have their clocks stopped.

38. Any one of claims **1**, **4–15**, **18–21**, or **24–37**, wherein said processor is a central processing unit (CPU).

* * * * *